Acquisition and Production of Skilled Behavior in Dynamic Decision-Making Tasks

Status Report

July 31, 1992

for NASA Ames Research Grant NAG2-656

Technical Monitor:

Robert J. Shively
NASA Ames Research Center
Mail Stop 262-3
Moffett Field, CA 94035

Principal Investigator

Alex Kirlik
Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

Overview

This status report describes work completed during the period January 31, 1992 to July 31, 1992. The following report describes two research projects. First, an ecological task analysis of the Star Cruiser task was performed and documented. On the basis of this analysis, a number of interface deficiencies were identified. As described in the January 31 status report, these deficiences will be eliminated through two alternative means: display enhancement and decision aiding. These two forms of aiding will be subsequently compared.

At this point in time, a new display for Star Cruiser has been constructed on the basis of the results of the ecological task analysis, as described below. In essence, the enhanced display provides perceptual support for actions for which the original display provided only meager or no support. An experiment is currently being performed comparing the efficacy of the original Star Cruiser display with the new, perceptually enhanced display. This experiment uses eight subjects in each display group, and is scheduled to run for sixteen ten-minute sessions for each subject. Each session uses a different initial conditions file. In the following, we present empirical data on the first six sessions of this experiment (the first six sessions beyond two half-hour training sessions).

The preliminary results are unequivocal. Performance in the perceptually enhanced display condition averages 35% higher than performance in the original display condition (performance measure is points scored). As can be seen by the enclosed graph of the results, the learning curves for the group means show no overlap over these sessions. It will be interesting to see whether this difference is maintained, increased, or attenuated, over the following ten experimental sessions.

The second research effort described here is completed work on modeling skilled decision-making using neural network and genetic-algorithm machine learning techniques. This section of the report consists of a Masters Thesis by Ling Rothrock.

Ecological Task Analysis of Star Cruiser

An ecological task analysis was performed on the Star Cruiser simulation. Expert subjects were created by having them continually practice the game until it appeared their learning curve had reached a plateau (approximately 20 sessions). These experts were then videotaped while playing Star Cruiser. Verbal protocols and written questionnaires were also given to the subjects. Reviewing these three items for each subject revealed the strategies the experts were using. As a result, each task in the analysis is described in terms of how the strategies involved it, whether or not perceptual cues existed to trigger that bit of strategy, and whether or not cues existed to indicate the availability of the actions comprising the task. The observations made during this analysis suggest possible enhancements that may be incorporated into the simulation's display interface. (These have not been included in this document.) Once an enhanced interface has been created, additional subjects will be used to determine if it is better than the original. The quality of the display will be measured by the subjects' performances (indicated by their score and the number of errors committed) on the new display compared to other subjects' performances on the original.

Deploy Probe

A probe may be deployed at any time except for when the Star Cruiser is docked at the Star Base. Deploying probes has no effect on points or fuel consumption and can therefore be done without great effect on the system states. Subjects used the probes to obtain information concerning ninth orbitals and solar systems' supply of information.

When determining which solar system to go to, subjects would dispatch probes to those systems, usually those with the brightest suns, under consideration. They would then be allowed to view the planets in those systems and ascertain the amount of information available for collection in each. This strategy was often unnecessary since the brightest suns, by definition, contained the largest amounts of information. The only benefit it served was to help choose which system to go to amongst those with the brightest suns. Other factors often contributed to this decision as well, though. Proximity to the Star Cruiser, the closeness of neighboring solar systems (subjects' strategies involve collecting more data from a grouping of solar systems), and the remaining time in the session often served as determinants in deciding which solar system to enter first.

As mentioned, probes were also used to locate the ninth orbital around a sun. Subjects used their knowledge of the probe's orbital path to help locate the proper orbital for Star Cruiser. This was not necessary at all times since other cues exist to help find the proper orbital (refer to *Move Star Cruiser into Orbit*).

These strategies are driven by the subjects' desire to obtain additional information about the galaxy. There are no cues that the subjects perceive which cause them to perform this action. In other words, the need to perform this action is only inferred by the subjects, no information is given instructing them to do so. This method of learning about the galaxy is generally a waste of time, however. Perceptual cues currently exist to provide all the same information to the subjects (i.e., sun color, planets orbiting near the ninth orbital). As previously stated, the only time that probes may not be deployed is when the Star Cruiser is docked at Star Base. There are no perceptual cues, though, that inform the subjects of this. Only their inability to do so suggests to the subjects that a probe cannot be deployed at that time. Also, the only cue that exists to indicate that a probe may be selected for deployment is when the user is able to highlight one at the top of the screen. This poses a problem, however, when the Star Cruiser enters a solar system while a subject is attempting to perform this action. On two separate occasions,

with two different subjects, this resulted in the Star Cruiser crashing into the sun. The subjects selected a probe to deploy while the Star Cruiser was in the galaxy view. During this process, the cruiser drifted into a solar system. The subjects' initial, and only, reactions upon seeing the Star Cruiser drifting towards the sun was to apply a thrust to the cruiser away from the sun. The probe, however, was still selected for deployment. As a result, the only actions the subjects could do successfully would be either to pull a string from the Star Cruiser to the sun to deploy the probe or to unselect the probe. Since the subjects were concerned solely with applying a thrust to the cruiser, they did not realize that they had to perform one of the other actions first. As a result, the thrust was not applied to the Star Cruiser and it crashed into the sun. Though their inability to perform the desired thrust action serves as a cue, it is embedded to deeply within the structure of the interface to be of use. The subjects simply feel that the applied thrust was not great enough to overcome the sun's gravitational pull and they continually try the action again. Some other form of cue is required in this situation to indicate to the subjects that they are performing an action unsuccessfully and that they should attempt another. The cue must serve to enlighten the subjects that their intended action is one that is not readily available.

Overall, there is no special need for using perceptual cues to inform the user when to deploy a probe. Most information that can be gained by doing so is present at all times. Cues that exist, though, at the depth structure level, especially those which indicate that the user needs to unselect a probe to perform a thrust, need to be brought to the surface. In addition, perceptual cues should indicate to the user when it is and is not possible to deploy probes.

Recall Probe

A probe may only be recalled when the Star Cruiser is in orbit in the same solar system as the probe and the current view is of that system. If these conditions are met, then the user may select the probe and draw a line back to the Star Cruiser to recall it. Recalling a probe has no effect on fuel consumption or points and can be time consuming. The only benefit is that the user now has an additional probe which may be deployed. Therefore, if the user determines that enough probes are present on board the Star Cruiser to complete the session, then

there is no reason why a probe should be recalled at any time. This agrees with the strategies of most of the expert subjects. One subject would, however, recall a probe while Star Cruiser was waiting for the collection tools to finish gathering information from the planets. His reasoning was that since Star Cruiser has visited the solar system, the amount of information present will always be displayed on the sun in the galaxy view. Therefore, if there are few probes which may be deployed, since he is not performing any other actions while waiting for the collection tools, he can recall the probe and not waste any time. This was also dependent on his confidence of being able to locate the ninth orbital if he ever returned to the same solar system in the future. Of course, if all information has been gathered from the planets in that system, then there will be no need to ever return.

Though there are no perceptual cues on the surface which indicate that this action should be performed, the user may determine to do it based on the number of probes displayed along the top of the screen. Whether or not a user will recall a probe will ultimately depend on whether or not it is felt that it will be used again. Currently, only the user's strategy of the session, including the time remaining and where the cruiser should be sent to in that time, will contribute to the decision to perform this action. As with the subject discussed previously, there are those who do always recall the probes though, if, for nothing else, to practice the task of recalling an object. Because of this action's insignificance to the overall goal of the user (collect information and return it to the Star Base), as well as everyone possessing a different strategy of the current session being played, it is probably unnecessary to incorporate cues which directly inform the user that a probe should be recalled.

Perceptual cues which inform the user when this action can and cannot be performed are also lacking at the surface. Though subjects had very little difficulty with determining this, there does exist the potential for some confusion. Since a probe may be deployed at almost any time, the risk is present that a user may think that it can be recalled at almost any time. This line of reasoning is further supported by the fact that in the galaxy view, the deployed probes are pictured next to their appropriate solar systems. Though this bit of information may be helpful in determining where the probes are located, it can also cause the user to adopt the wrong retrieval strategy. To avoid the mis-specification between the cue (displayed probes) and a possible action it may indicate (can recall the probes), the cue should either be altered or eliminated. This is so even though the

user's inability to recall a probe may indicate that s/he is using an improper retrieval strategy. The key here is not to correct the user once the mistake is made, but to prevent the mistake from being made in the first place.

A user may relate the retrieval of a probe to that of a collection tool. This would then serve as a cue to the user: since a tool may only be recalled while the Star Cruiser is in orbit, the same theory may be devised for the probes. Though this would be correct, the users should not be subjected to the burden of rationalizing this for themselves. This similarity between recalling the tools and the probes should be evident from the display itself. Taking all into consideration, for the display to contain the proper cues, it must not only inform the user of when the probes can be recalled, but it must not mislead the user into thinking they can be when they truly cannot.

Deploy Collection Tool

Only under certain conditions may a user deploy one of the four collection tools: satellites, robot miners, science ships, and minerships. The Star Cruiser must be in orbit around a sun which also has orbiting planets. In addition, the user must be viewing that system. If the user wishes to deploy either a science ship or a minership, then planets which support life, green planets, must also be present in the solar system.

Each subject possessed a different strategy for determining where and when to deploy collection tools. One subject, for instance, would first send out science ships and minerships whenever possible before satellites and robot miners. Her strategy was to collect as much information as possible as quickly as possible. She viewed the collection of data and resources using the science ships and minerships to be quicker since she would not be required to make as many deployments. She, more than any other subject, appeared to have the greatest difficulty with performing the necessary tracking task required to deploy the collection tools. Another subject would normally deploy only satellites and robot miners since these did not move from planet to planet and she was thus better able to remember how much information they had collected. Only if one green planet was present would she then send out a science ship or minership. This strategy was developed in order to reduce the risk of a science ship or minership collecting to much information and thus overloading the cruiser when it is

recalled. The subject would also deploy as many tools as she had available, even if the cruiser would not be able to carry all of the information. It was her reasoning that she could always return to the system after unloading the Star Cruiser at the Star Base and recall the remaining tools. A third subject had yet a different strategy for deploying collection tools. He, like the first subject, would deploy science ships and minerships first. Unlike the first subject though, this subject would send out multiple science ships and minerships to speed up the collection process even more. This subject differed from the other two in that, after recalling the other tools, he would deploy satellites and robot miners to other planets to collect a fraction of their data and/or resources. His strategy focused on returning to the Star Base with the Star Cruiser as fully loaded as possible whereas the other subjects were content with the cruiser not being as full. He usually did not leave tools in the solar system once he had the cruiser leave. There did seem to be one bit of strategy that all subjects had in common though. In deciding where to send a collection tool, the subjects generally did not send more than one data collecting tool (satellite or science ship) or resource collecting tool (robot miner or minership) to the same planet.

The fact that all subjects had differing strategies serves as evidence to the lack of perceptual cues that exist for aiding the users in deciding when and where to deploy the collection tools. If these cues were present to indicate when to perform the action, then there would most likely be more similarities between each subject's strategy. This is illustrated by the subjects' common attitude that if a tool is already collecting one type of information from the planet, then no other tool which collects similar information should be deployed to that planet. A cue does exist for this even though it is informing the subjects not to perform the action rather then to do so. When a collection tool is deployed to a planet, it is displayed next to it. Therefore, the user can see which tools have been sent to which planets. As a result, the user knows that the deployment of any other tool to that planet (which collects similar information) is essentially pointless since the tool presently there will collect all of the data or resources that the planet contains. The only benefit that deploying another tool which collects the same type of information to that planet would possibly serve is that one tool will not be transporting to much to the cruiser, and thus overload it, once it is recalled. This never seemed apparent to the subjects since they were not observed performing an action of this type. This is mostly likely due to the lack of cues which would

inform them of how much each tool has collected, let alone whether this action was even possible.

None of the subjects appeared to have difficulty determining when the deployment of tools could be performed. The subjects were instructed during their training about the conditions that must be met in order to perform this action. With this knowledge, they were able to recognize easily when the conditions were satisfied. Perceptual cues, such as the Star Cruiser being highlighted upon reaching orbit, the absence of the cruiser and/or the planets when the subjects were looking at a view of the galaxy or another solar system, and whether or not any planets were present in that solar system at all, also provided the necessary support in determining when the action could or could not be performed. The fact that the tools were always present at the top of the screen, thus appearing as if they could be selected even when they could not, did not cause the subjects any difficulties. If, for some reason, they tried to select one of the tools when that action could not be performed, they would soon discover that their attempt was not allowed due to their failure to do so. Overall, the cues were successful at specifying the availability of the action.

There was, however, one area of underspecification. Though they knew they could perform some type of deployment, the subjects had problems determining which tools should be used to collect which information and which type of planets they could be sent to. Most of the subjects eventually memorized the differences between the tools. There was one subject though, that had difficulty throughout the sessions. The tools themselves provide no cues as to what they do. They are not coded in any fashion whether it be, for example, color, size, or location (at the top of the screen where they may be selected). Some form of cues should be incorporated into the display in an effort to assist the user. Successful implementation of such cues should result in the user's reduced confusion concerning each type of tool's functions and constraints as well as reducing their need to rely on memory to determine them.

Deploying collection tools was consistently rated as one of the more difficult actions to perform. Though much of this is attributed to the tracking task involved, some of the blame can be placed on the amount of mental effort the subjects had to use in order to accomplish the action. This is suggested by this action being rated slightly more difficult then that of recalling a tool which essentially incorporates the same type of tracking task. The subjects had to remember which type of planet each tool could be sent to as well as what each

collected. Even more difficult though, they had to determine exactly which planet to send a tool to. This often required the subject to take into consideration the amount of data/resources available on that planet (indicated by the "pie pieces" displayed on each), the amount currently on board the Star Cruiser, the amount of data/resources being currently collected by other tools, whether the deployed tool would move from planet to planet, and how much time was left in the mission. Though much, but not all, of this information was presented to the subjects in one form or another, they were required to interpret each bit and relate each piece of information to the others in order to make a decision. This was often quite complex and thus it is understandable why this action was rated to be so difficult. The proper use of present perceptual cues, in addition to the introduction of new ones, should help to make this action an easier one for the users to perform.

Recall Collection Tool

In order to recall collection tools, similar criteria as that for deploying the tools must be satisfied. This includes viewing a solar system where the Star Cruiser is in orbit. The only difference in the criteria is that, in that system, tools, deployed earlier, must still be located at the planets.

Subjects generally used similar strategies in determining when it was, and was not, appropriate to recall a collection tool. The subjects would recall a tool if two conditions were satisfied. The first is that the tool had finished collecting all of the data/resources that were available. The other was that there was room aboard the cruiser to carry the collected information. The remaining time left in the mission also played a role in determining if a tool should be recalled or abandoned so that the cruiser could return to the Star Base with what was already on board.

The subjects had very little difficulty in determining when they could perform this action. Cues were present and noticeable to indicate when a tool could be recalled. As in deployment, the highlighting of the Star Cruiser once it is in orbit informs the user that that portion of the criteria has been satisfied. In addition, the subjects knew which tools could be recalled due to them being displayed next to a planet in the solar system. Confusion does exist, however, in determining which tool will be recalled first when multiple tools are present at

the same planet. Since their displayed icons overlap, there is no apparent cue to indicate which tool will be recalled first when the user selects one. In order to overcome this, the subjects were instructed that the order of recall was identical to the order, from left to right, of collection tools displayed at the top of the screen. It was observed though, that even with this knowledge, the subjects would attempt to recall one tool from a planet containing multiples, and inadvertently recall the wrong one. As a result, they would usually recall the correct tool and then deploy the one that was incorrectly brought back in order to finish collecting the available information at that planet. This problem could be avoided by simply locating each type of tool in a different position around the planets. With this exception, the availability of this action was generally well perceived by the subjects from the existing perceptual cues.

The ease in deciding whether or not to perform this action was variable throughout the mission. It greatly depended on the second of the two criteria mentioned earlier - how much data/resources the Star Cruiser currently had on board. If the cruiser was empty, then sufficient perceptual cues existed to suggest to the user that s/he recall a tool. The fact that the tool had completed collecting all possible information would be indicated by the disappearance of the "pie pieces" which show how much data/resources is present on the planet. Since the gauges which convey how much information the Star Cruiser contains would be empty, the user would know that recalling a tool was more than likely an appropriate action to perform. As the Star Cruiser contained more and more information though, the decision to recall a tool became more and more difficult. Even though the user could still readily determine if the tool had completed its collection task, insufficient cues existed to inform the user whether or not the information collected could be safely loaded onto the cruiser. The problem is that no direct relationship exists between how much of the information, represented by the pie pieces, is collected and how much the gauges indicating the cruiser's load will increase once those tools are recalled.

Unless it was learned exactly how many pie pieces completely loaded the cruiser (which two subjects did over ten sessions), the user would never be quite sure how much additional data/resources can be brought onto the cruiser before it exceeds capacity. Even when it was learned that, for example, three full planets completely loaded the cruiser, difficulties still arose when the subjects were forced to deal with planets which contain only a fraction of their total capacity of information. This problem was compounded further by the pie pieces for two

reasons. One is that the same size pie piece represented a range of amount of data or resources. For example, a half of a pie piece (one full pie piece equals one half the size of the planet) of data could represent between one-fourth and one-half of the planets capacity for a type of information. There is no way to determine the exact amount. Contributing even more to the problem is that the gauges indicating the cruiser's current load merely present qualitative information - how full is the Star Cruiser. As a result, it is also difficult to determine exactly how much data/resources the cruiser already contains as well as how much the gauges will increase by collecting a certain size pie piece. The second reason is that the pieces will disappear as the information is collected. Though this serves as a good cue indicating when the collection process is complete, unless the user remembers how big of a pie piece was present before the tool was deployed to the planet, there is no way for the user to know how much data/resources the tool has collected. This is especially true for science ships and minerships which can visit multiple planets before being recalled. This is an even greater problem when the user has done something (i.e., returned the cruise to the Star Base) which results in the viewing of the galaxy or another solar system.

The goal of the user is to collect as much information as possible and return it to the Star Base. As a result, this action of recalling collection tools is one of the most crucial. The users' success at completing their mission ultimately depend on their ability to make good decisions concerning when and when not to recall a collection tool. The display should be designed to aid the decision-making process, not hinder it. Therefore, enhancements should be made to the current display in order to improve the users' chances of success. Creating gauges and pie pieces which have direct relationships is one such improvement. Others may include memory aids which will help the users remember exactly how much information has been collected by a tool. Warnings can also be incorporated indicating when the Star Cruiser has reached near capacity or even when the recall of a particular tool will overload it. These are just some of the possible enhancements which can assist the users in deciding when to recall a tool. As a result, not only should the users' performance of this action improve, but so should that of their overall mission as well.

Place Star Cruiser Into Orbit

The Star Cruiser will achieve orbit around a sun if it passes through the sun's ninth orbital at a slow enough speed. This of course means that the user must place the cruiser in the solar system view which contains the sun to be orbited.

Subjects shared a similar reason for wanting the Star Cruiser to obtain orbit - to deploy collection tools. Unless they had taken the cruiser out of orbit while tools were collecting information, the subjects generally did not try to place the ship into orbit for the sole purpose of recalling collection tools. This also is true for probes. If a subject made the decision to recall a probe, it was only while the Star Cruiser was already in orbit.

Only previous instruction lets the user know that s/he must obtain orbit. Since it is known that the Star Cruiser must be in orbit in order to perform any actions regarding the collection tools, once the decision has been made to deploy or recall a tool, the user knows that the cruiser must be placed in orbit. Thus, in a sense, those cues which aid the user in deciding whether or not to deploy or recall tools (i.e., presence of pie pieces on planets; absence of planets) also serve as cues to put the ship into orbit. There is, however, no direct mapping between the desired action of deploying or recalling a tool and the necessary means for doing so such as first obtaining orbit. As a result, those users who do not receive instructions prior to attempting a mission may try to deploy a tool while the cruiser is not in orbit. Of course, their failure to complete this action will indicate to them that something is wrong, but they would most likely be unable to determine what.

Assuming that the user knows that the Star Cruiser must be in orbit, very few perceptual cues are required to indicate the availability of the action. Knowing that the ninth orbital must be located, the user generally realizes that only in a view of a solar system may this action be attempted. Thus, once the cruiser moves into a view of a solar system, no other requirements must be met in order to attempt this action.

The difficulty concerning this action, according to the subjects, was not in determining when to perform the action nor if the action could be performed. It was in performing the action itself that presented the most trouble. The subjects often complained about how hard it was to locate the ninth orbital, let alone get the Star Cruiser moving at the proper unknown speed so it would follow the orbital around the sun. As a result, they were often required to continually adjust the cruiser's direction and speed until the cruiser reached orbit. A frustrating

process which often wasted valuable time. Regarding the speed, there are no cues whatsoever which would indicate to the user that the ship has the proper velocity for obtaining orbit. Only upon seeing the cruiser obtain orbit (Star Cruiser is highlighted) will the user have any idea of what the proper speed is. The user, however, usually has difficulty remembering what the speed was, or duplicating it, since s/he is required to match it to a pictorial representation of the cruiser's movements. No quantitative information is provided. There are several cues, though not intuitive, which may be utilized in locating the ninth orbital around a sun. The first is to use the planets as a guide. Eventually it should be learned that the furthest orbital for a planet is one less then that of the cruiser. This allows the user to approximate the ninth orbital's location based on the orbitals of the existing planets. Another cue that may be learned over time is that the orbital is elliptical in shape and that the top of it is aligned with the top of the solar system view. The user can then "picture" the path that the orbital takes starting at the top of the view and attempt to have the cruiser intercept that pictured path. A third method for finding the orbital, one consistently performed by several of the subjects, is to dispatch a probe in the solar system where the cruiser is to obtain orbit. Once again something that is learned through practice, the users soon realize that the probe travels around the sun in the ninth orbital. This "trick" allows the user to deploy a probe to identify the orbital which can then be easily located while attempting to place the cruiser. So though cues do exist to show where the orbital is located, they are not readily perceived by the user unless their existence has been learned.

While the amount of perceptual cues indicating that this action should be performed and that it can be might be sufficient, additional cues should be added to assist the user in placing the Star Cruiser into orbit. To be effective, the cues would need to alert the user to the location of the ninth orbital. This can simply be accomplished by highlighting the orbital in some manner. In addition, some form of a cue should be used to indicate when the cruiser is at the required speed for obtaining orbit. This may include, for example, the use of a "speedometer" or changing the color of the cruiser. Without these additions, users will face similar troubles and suffer from similar frustrations as they attempt to place the Star Cruiser into orbit.

Remove Star Cruiser From Orbit

This action may only be performed if the Star Cruiser is, obviously, in orbit in some solar system and the user is viewing that system. Subjects usually took the cruiser out of orbit only if one of two conditions was met. If all possible information had been collected in the system, and all tools had been recalled, then the subjects removed the cruiser from orbit in order to send it to another solar system or to the Star Base. On the other hand, if more information could still be collected but there was no additional storage space on board the ship, then the Star Cruiser was taken out of orbit as well. In this case, the cruiser was returned directly to the Star Base.

The number of perceptual cues which would lead the user to perform this action are minimal. When presented a solar system view where no tools or probes are deployed and the planets, if there are any, no longer contain any information as indicated by the pie pieces, then the user should realize that there is no reason to have the cruiser in orbit. Others cues may also be present which may lead to the decision to remove the Star Cruiser from orbit. These, however, do so indirectly since they actually inform the user that another action should be performed (refer to *Dock Star Cruiser at the Star Base).* In order to perform this action though, the user must remove the cruiser from orbit first. As a result, although the means are slightly different, the results they obtain are similar. In this case, this is satisfactory since the relationship between the two actions (one must precede the other) is an easy one for the user to make. The action of removing the ship from orbit is easily triggered by the cues which currently exist and, thus, no modifications are necessarily required.

As mentioned before, the user must be viewing the solar system which contains the orbiting Star Cruiser in order to perform this action. If not, then the user's inability to successfully remove the cruiser from orbit serves as a cue that the action cannot be performed. Otherwise, there are no other cues which indicate this to the user, nor that the user can perform the action. It was noted, though, that subjects very rarely, if ever, attempted to remove the Star Cruiser from orbit when it was impossible to do so. In addition, whenever they decided to perform the action, they were in a situation which permitted the action. Thus, it appears that whenever the user's strategy calls for removing the cruiser from orbit, no additional cues are required to show the availability of that action.

Problems do arise, however, when the user does not wish to remove the cruiser from orbit, but does so accidentally. This is a result of the aforementioned

lack of cues which indicate that this action may be performed. Several attempts made by the subjects to deploy tools resulted in their accidental removing of the Star Cruiser from orbit. When deploying a tool, after selecting it from the top of the screen, the user is required to select the Star Cruiser as it is travelling around the sun. If the user is unable to track and select the ship properly, then the tool is unselected. The subjects would not realize this and attempt to select the cruiser again. If successful at the second attempt, since the tool is no longer selected, they would actually be placing a thrust on the cruiser. This more than likely pulled the ship out of orbit at an undesirable time. This error was often experienced during the earlier sessions. Even with more practice, the mistake was still made, though not as frequently. The subjects learned that they needed to select the tool again before selecting the cruiser. Some form of cue should be present, however, to indicate to the user when a string from the cruiser will deploy a tool or probe or when it will act as a thrust. In other words, a perceptual cue should be implemented which will inform the user of the availability of this action.

Dock Star Cruiser At Star Base

The Star Cruiser can be docked at the Star Base if it is moving around in the galaxy and the user is viewing it. The cruiser must be traveling at a slow enough speed as it passes the base in order for it to dock.

Subjects shared similar strategies in deciding when to dock the cruiser at the Star Base. Any one of three situations would result in the subjects abandoning their current activities and performing the necessary actions that would lead to the cruiser's docking. The main reason people docked the cruiser was when they wished to unload its contents. Some subjects attempted to fill the cruiser as full of information as a solar system's planets would allow, while others were satisfied with partial fillings. Another factor which often led to subjects moving the ship to the Star Base was the time remaining in the session. If the cruiser had some information on board and time was nearly expired, the subjects would dock the cruiser as quickly as possible in order to increase their point total. The third factor which compelled people to perform this action was the amount of fuel the cruiser had remaining. Lack of fuel would force subjects to return the cruiser back to the Star Base in order to refuel. Essentially, as long as

the cruiser had enough cargo space and fuel, the subjects were content with keeping it moving through the galaxy and collecting information from the various solar systems.

The availability of this action was usually not questioned by the subjects. Though not often attempted, an inability to maneuver the Star Cruiser while it is located somewhere other than the current view provides the necessary cue to the user that the action cannot be performed. Due to the nature of this action though, this is not often enough. In order to dock the cruiser at the base, the user may have to, or wish to, perform other actions first. For instance, if the Star Cruiser is in orbit, the user must take it out of orbit before s/he can dock it at the Star Base. Therefore, in some cases, the availability of this action depends on that of others and thus the cues which signify the availability of those other actions become significant. The subjects did not seem to have trouble with this aspect of the docking action. They took into account such relationships between actions and performed them accordingly. This does not infer though, that the difficulties which existed for the other actions are no longer. The user still runs the risk of experiencing those troubles resulting from inappropriate cues that were discussed under each of those individual actions.

From the subjects' performances, there appeared to be several perceptual cues which led them to perform this action. This is especially true when considering the cargo space and time factors. The gauges depicting the amount of cargo space aboard the Star Cruiser were often referred to during the collection of information. As a result, the status of the ship's cargo space was often known. Thus, if the user deemed it necessary, the Star Cruiser would be returned to the Star Base. The presence of the timer during the mission provided the subjects with the necessary information concerning the time remaining. It too was often checked to determine if the Star Cruiser should be returned to the Star Base, or if any other actions should even be attempted.

The subjects often ran into difficulty when it came to the third factor. The Star Cruiser's fuel level was very rarely monitored by the subjects. As a result, the cruiser would explode, thus ending the mission, much to the surprise of the user. This happens even though there exists a fuel gauge monitoring the amount remaining in the cruiser. A possible explanation for this is that while the cruiser is in orbit, the subject is solely concentrating on the collection of information. Thus, the only gauges of concern are the cruiser's storage capacity gauges. In addition, since no thrusts are being applied to the Star Cruiser, no

fuel is consumed and therefore there is no need to monitor the fuel gauge. If the cruiser is not in orbit, then the subjects appeared to be more concerned with steering the ship through the solar systems and galaxy in order to avoid it from crashing into any suns and/or to make sure sure it is heading back to the Star Base. Very little concern is placed on the fuel consumption gauge. Thus, if only a small amount is present, since the subjects were likely not to notice, after performing several thrusts, the ship would explode. This was very noticeable whenever the subjects had difficulty controlling the speed of the cruiser. Since no cues exist which convey this information to the user (refer to *Place Star Cruiser Into Orbit)*, it sometimes took several attempts to get the cruiser travelling at the proper docking speed. This of course may only involve small thrusts, but they can be numerous. As a result again, the ship will run out of fuel and explode.

This is a problem resulting from the perceptual cues' overspecification of the action. Too many cues exist independent of one another. The user is unable to efficiently monitor, identify, or notice all cues. Therefore, the same amount of information that is available to the user should somehow be restructured. In doing so though, no additional effort, mentally or physically, should have to be exerted by the user in order to successful determine when to dock the Star Cruiser at the Star Base. One possible method for accomplishing this is to reduce the amount of independent cues. For example, combining the information presented by multiple cues into one cue should help the user in determining when to perform this action. Whatever the method though, care must be taken that the modification does not result in the action becoming underspecified, mismatched, or even more overspecified than before.

Release Star Cruiser From Star Base

This action was rated by the subjects as the easiest which involves movement of the Star Cruiser. Not only is it easy to perform the action, applying a thrust to a stationary object, but determining when to perform the action is simple as well. Unless the amount of time remaining in the mission is so low that nothing can be accomplished or all information within the galaxy has been collected, the subjects would remove the Star Cruiser from the Star Base immediately after it had docked.

Since the users cannot perform any other actions that affect the status of the galaxy while the cruiser is docked at the Star Base, they understand that in order to continue achieving their goal of collecting information, they need to pull the cruiser away from the base and have it travel towards one of the solar systems. This serves as a forcing function which guarantees the users performing the action if they are going to better their performance. In a sense then, the requirement to remove the cruiser from the base in order to affect the status of the galaxy is enough to inform the users that they should perform this action. Therefore, no additional perceptual cues are required.

The only time this action would not be available to the user is if the current view is of a solar system. Only when viewing the galaxy is the user capable of removing the Star Cruiser from the Star Base. In a view of a solar system though, the cruiser would not even be displayed. Thus, the absence of the cruiser signifies to the user that a thrust cannot be applied to the ship to move it away from the dock. Whenever the user is viewing the galaxy, however, s/he may easily apply the thrust and pull the Star Cruiser away from the base. Since it was apparent that the subjects consistently knew about the availability of this action, no further perceptual cues are warranted.

Change View To Galaxy / Solar System

Whenever subjects performed this action, it was to gather information which would help determine the Star Cruiser's next movement. With the cruiser in the galaxy, subjects would select a view of a solar system to assess the amount of information available on its planets. Different solar systems would be selected until the subjects determined that one contained enough information to justify sending the cruiser there. The cruiser's next movement would then be towards that particular solar system. If the cruiser was present in the solar system, most always in orbit, the subjects often selected the galaxy view in order to determine which side of the system the cruiser should exit from. The subjects who performed this action for this reason found it necessary to do so since the thrusts they applied to the cruiser to break it out of its orbit often sent it flying out of the solar system uncontrollably. Information obtained from viewing the galaxy, such as the location of neighboring solar systems or the location of the Star Base, often helped the subjects to either prevent the cruiser from sailing into another solar

system and crashing into it's sun or find the shortest route for the cruiser to return to the Star Base or to travel to another solar system. Often these alternate views are selected while the subject is waiting for the completion of another action (i.e., tools collecting information, cruiser traveling through galaxy, cruiser docked at the Star Base).

Though some subjects successfully employed these strategies, some do not. Even though they were told about the option of different views and how to access them, some subjects would forget of its existence. This is because no perceptual cues are present to inform the user of the action's availability, even if the user remembers it from training. There is nothing about the display which lets the user know that by selecting a sun in the galaxy view, or the sun in the solar system view, s/he can observe a different map. Though there are cues which indicate which solar systems may be viewed (the presence of the pie pieces on the suns), these still do not inform the user that the action can be performed.

Additionally, cues which assist the user in determining when to perform this action are lacking. Only as a result of practice and habit does the user incorporate this action into a strategy. And then again, the multiple views are only employed if s/he fails to remember the various states of the system. As a result, this action currently does little to help relieve the memory burden placed on the user.

There is essentially no perceptual cues which specify this action. The subjects who did perform it though, generally had an easier time determining where to send the Star Cruiser. They also seemed more capable of getting the ship there utilizing the smallest amount of effort. Therefore, the action has merit and the environment should support it. In order to do this, perceptual cues need to be introduced which will aid the user in determining if and when to perform this action, as well as whether or not the action can be performed.

Summary

The movement of the Star Cruiser, as controlled by the users, is determined by a number of factors. These factors include the distance the ship has to travel, the configuration of solar systems within the galaxy, the cruiser's amount of fuel, and the time remaining in the mission. They all play a vital role in the determination of where the Star Cruiser needs to go and how it is going to get

there. All of this information is readily available to the user via the display interface. In addition, they all serve as cues which contribute to the decision to perform other actions. Quite often, the user elects to perform an action but cannot, or will not, do so until another is performed. Many of the actions are related through such a hierarchy. For example, a subject may elect to remove the Star Cruiser from orbit. As a result, s/he might first perform the action of viewing the galaxy to determine the cruiser's destination and exit path and then actual remove the cruiser from orbit. Such relations can be as simple as this, or they may be more complex. The action of returning information to the Star Base may involve the user first deploying a probe, viewing the solar system containing that probe, steering the cruiser into that system, obtaining orbit, deploying collection tools, recalling the tools, removing the cruiser from its orbit (which itself may require multiple steps), steering it through the galaxy, and finally docking it at the Star Base.

The user must clearly understand these relations if any acceptable performance is going to be obtained. The perceptual cues throughout the environment play a significant role in this process. The cues should accomplish two goals. The first is that they should always let the user know when a certain action may be performed. Secondly, if an action is unavailable, then cues should specify to the user what actions need to be performed in order to make the unavailable one possible. This process should repeat until the user is presented with an available action that s/he knows will help achieve the availability of the original action. Whenever breakdowns is this process occur, where the user does not readily know what to do next or if it even can be done, then the existing perceptual cues should be investigated and the determination made about whether new cues should be used, current cues should be eliminated, or multiple cues should be combined.

The preceding discussion concerning Star Cruiser's possible actions attempted to accomplish this. The strategies of expert subjects were examined to determine when a particular action was likely to be performed. The subjects' abilities to perform these actions were then evaluated. Possible explanations for the ease or difficulty of each, based on the perceptual guidance supplied within the Star Cruiser environment, were also offered. As a result, areas of improvement can now be suggested. These improvements should concentrate on improving Star Cruiser's environment so that, where it failed before, the

perceptual guidance afforded will specify what actions the users wish to do as well as whether those actions can be performed.

Task Analysis Results: Enhancements

From the ecological task analysis performed on the following actions, certain factors were discovered that warranted possible enhancements in order to improve user performance. Listed for each action are those factors along with suggested enhancements based on perceptual cues and decision aids.

Choose Star Cruiser's Destination

1) cruiser's amount of fuel

Perceptual Cue Enhancements:

- Fuel gauge will be modified to make fuel level more noticeable.

Decision Aid Enhancement Features:

- The aid will indicate whether the cruiser needs to return to star base in order to refuel.
- A ranked list of possible destinations for the cruiser will be indicated to the user (through color coding or messages in a window area) based partially on whether the cruiser has enough fuel to fly to that destination from its present location and then fly to the star base.

2) amount of information contained within solar system

Perceptual Cue Enhancements:

- Color of sun will indicate amount of information contained within solar system relative to amount of information contained within galaxy.
- Use "Ghost Images" on suns in galaxy view to depict amount of information once present in system (user will then know if cruiser has previously collected information from that system).

Decision Aid Enhancement Features:

- A ranked list of possible destinations for the cruiser will be indicated to the user (through color coding or messages in a window area) based partially on amount of information contained within each solar system.

3) cruiser's remaining information storage capacity

Perceptual Cue Enhancements:

- Gauges indicating cruiser's storage capacity will be altered to resemble those depicting the amount of information contained on a

planet (pie charts). User can then compare shape of pie piece on planet to empty space in pie gauges representing storage capacity to determine if amount of information will fit onto cruiser. (Simpler to incorporate then polygon-changing display.)

Decision Aid Enhancement Features:

- The aid will indicate whether the cruiser needs to return to star base in order to unload the information on board the cruiser.
- A ranked list of possible destinations for the cruiser will be indicated to the user (through color coding or messages in a window area) based partially on whether the cruiser has any remaining capacity for the type of information available within each solar system.

Collect Information
(Combination of Deploy Collection Tools and Recall Collection Tools)

1) tools allowed to deploy

Perceptual Cue Enhancements:

- Color coding tools selection bar to facilitate determination of which tools collect what and where they may be deployed to.
- Dim tools when they cannot be deployed

Decision Aid Enhancement Features:

- A ranked list of tools to be deployed will be indicated to the user (through a change of color of a message) based partially on whether that tool can be legally deployed in that solar system (for example, if there are no life sustaining planets in the solar system, no manned ships may be deployed).
- After a tool has been selected by the user, the aid can then indicate a ranked list of planets to deploy the tool to (through a change of color or a message).
- The aid will indicate whether the user is trying to deploy an inappropriate tool to an inappropriate planet (for example, a manned ship to a planet which does not support life, or a manned ship to a planet without any information).

2) amount of information contained within solar system

Perceptual Cue Enhancements:

- Color of sun will indicate amount of information contained within solar system relative to amount of information contained within galaxy.
- Use "Ghost Images" on suns in galaxy view to depict amount of information once present in system (user will then know if cruiser has previously collected information from that system).

Decision Aid Enhancement Features:

- If more information is contained within that solar system than can be collected by the cruiser, the aid will use this fact when indicating a

ranked list of tools to deploy, a ranked list of planets to deploy the tools to, and when indicating a ranked list of tools to recall.

3) cruiser's information storage capacity

Perceptual Cue Enhancements:

- Gauges indicating cruiser's storage capacity will be altered to resemble those depicting the amount of information contained on a planet.

Decision Aid Enhancement Features:

- If deploying a robot ship to a planet in a solar system could result in the tool collecting more information than can be stored on the cruiser (due to its automatically flying to other planets), this fact will be considered when the aid indicates a ranked list of tools to be deployed, a ranked list of tools to recall, and a list of planets to deploy tools to.
- The aid will consider the remaining capacity it has to store information when indicating a ranked list of tools to recall from planets.
- The aid will indicate whether the user is trying to recall a tool which would overload the star cruiser.

4) amount of information collected by tool

Perceptual Cue Enhancements:

- "Ghost image" of collected information so user knows how much tool will transfer to cruiser.

Decision Aid Enhancement Features:

- If deploying a robot ship to a planet in a solar system could result in the tool collecting more information than can be stored on the cruiser (due to its automatically flying to other planets), this fact will be considered when the aid indicates a ranked list of tools to be deployed, a ranked list of tools to recall, and a list of planets to deploy tools to.
- The aid will consider the amount of information collected by each tools (in relation to the remaining capacity on board the cruiser) when indicating a ranked list of tools to recall from planets.
- The aid will indicate whether the user is trying to recall a tool which would overload the star cruiser.

5) types of tools deployed to each planet

Perceptual Cue Enhancements:

- Locate deployed tools around planet, one tool type in each corner. (Ease of implementation questionable.)

Decision Aid Enhancement Features:

- When indicating a ranked list of tools to deploy and a list of planets to deploy the tools to, the aid will consider what tools have already been deployed to that particular planet.
- Because the tools may only be recalled in a specified order (see Star Cruiser specifications), the aid will consider this order when indicating a ranked list of tools to recall from a solar system.

Place Star Cruiser Into Orbit

1) location of ninth orbital

Perceptual Cue Enhancements:

- Highlight ninth orbital.

Decision Aid Enhancement Features:

- Hints could be suggested to the user in a window, such as "Deploy probe to find ninth orbital" or "Outer planet is in the seventh orbital."

OR

- Advice could be provided to the user indicating which direction to move the star cruiser to obtain orbit (such as placing arrows by the star cruiser, indicating the direction).

2) cruiser's speed

Perceptual Cue Enhancements:

- Change color of cruiser when it has obtained orbiting speed (or slower).

Decision Aid Enhancement Features:

- Hints could be suggested to the user in a window, such as "You have reached orbital speed."

OR

- Advice could be provided to the user indicating whether or not the cruiser needs to slow down or has reached an acceptable speed (for example, from the previous section the arrows placed by the star cruiser could turn different colors, depending upon the current speed of the cruiser).

Controlling Star Cruiser's Movements and Deploy Probe

1) function of string pull upon cruiser

Perceptual Cue Enhancements:

- Change color of cruiser to indicate type of action string pull will have on cruiser.

Decision Aid Enhancement Features:

- The aid could indicate to the user which type of action a string pull will have on the cruiser.

Remove Star Cruiser From Orbit

1) amount of thrust required for cruiser to break orbit

Perceptual Cue Enhancements:

- Flash cruiser when thrust to be applied to cruiser (as indicated by string pull) is enough for it to break orbit. (Questionable as to whether or not this is really needed.)

Decision Aid Enhancement Features:
- No decision aid enhancement features are proposed here.

Dock Star Cruiser At Star Base

1) cruiser's amount of fuel
Perceptual Cue Enhancements:
- (See **Choose Star Cruiser's Destination**)

Decision Aid Enhancement Features:
- (See **Choose Star Cruiser's Destination**)

2) amount of information stored on cruiser / cruiser's remaining storage capacity
Perceptual Cue Enhancements:
- (See **Choose Star Cruiser's Destination**)

Decision Aid Enhancement Features:
- (See **Choose Star Cruiser's Destination**)

3) availability of information in other solar systems
Perceptual Cue Enhancements:
- (See **Choose Star Cruiser's Destination**)

Decision Aid Enhancement Features:
- (See **Choose Star Cruiser's Destination**)

4) speed of cruiser
Perceptual Cue Enhancements:
- Change color of cruiser when it has obtained docking speed (or slower).

Decision Aid Enhancement Features:
- When docking at star base, the aid can indicate whether the cruiser has obtained docking speed or needs to slow down (through a change in color or a message).

Changing Views

1) may only perform if cruiser has previously visited solar system or if a probe has been deployed to it
Perceptual Cue Enhancements:
No perceptual cue enhancements proposed due to complexity of changes that would be required.

Decision Aid Enhancement Features:
No decision aid enhancement features are proposed here.

Recall Probe

No proposed enhancements since no common factors between perceptual cue and decision aid enhancements.

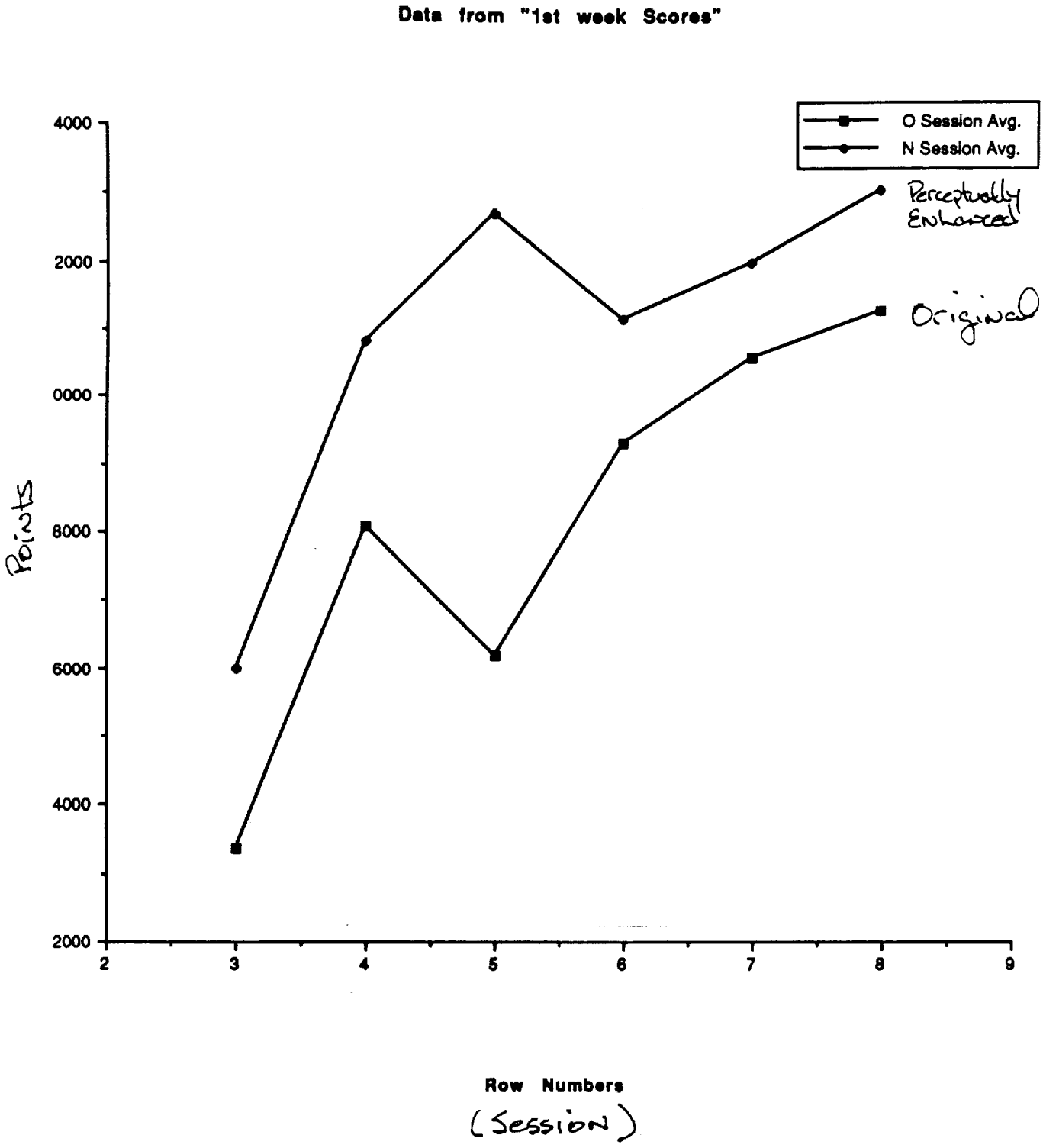
Release Star Cruiser From Star Base

No enhancements required. None were deemed necessary.

Experimentation

At this point, the perceptually enhanced version of the display has been constructed, and building the decision aiding system is in progress. An experiment is currently being performed to compare learning and performance using the original display with learning and perfomance using the perceptually enhanced display. Eight subjects are being run in each display group for a total of 16 sessions (after two one-half hour training sessions).

Initial results, shown on following pages, are encouraging. On average, subjects in the enhanced display condition perform at a level 35% higher than the subjects using the original display.


Data from "1st week Scores"
O Session Avg.
N Session Avg.
Perceptually Enhanced
Original
Points
4000
2000
0000
8000
6000
4000
2000
2
3
4
5
6
7
8
9
Row Numbers
(Session)

1st week Scores

Tue, Aug 4, 1992 2:29 PM

	Euwanda	Joseline	Kelli	Richard	David	Ritchie	Bill
1							
2							
3	5488.000	6930.000	840.000	0.000	0.000	0.000	5544.000
4	5278.000	6048.000	5887.000	12005.000	9079.000	5418.000	9177.000
5	7805.000	5530.000	3528.000	10108.000	10031.000	0.000	0.000
6	7371.000	7049.000	7049.000	15372.000	0.000	10437.000	15400.000
7	10430.000	4480.000	5544.000	16541.000	11088.000	11487.000	13314.000
8	11571.000	10514.000	10311.000	13132.000	15967.000	16842.000	11494.000

1st week Scores

Tue, Aug 4, 1992 2:29 PM

		Jamie	O Session Avg.	Natalie	Jacqui	Spring	Scott B	Scott Gu
	1							
	2							
	3	8120.000	3365.250	3416.000	7035.000	8757.000	5614.000	6930.000
	4	11802.000	8086.750	7672.000	6811.000	10577.000	15988.000	15561.000
	5	12383.000	6173.125	7245.000	10780.000	9296.000	16387.000	11011.000
	6	11529.000	9275.875	9506.000	7672.000	6853.000	13342.000	12880.000
	7	11480.000	10545.500	10346.000	11109.000	10787.000	19558.000	12628.000
	8	0.000	11228.875	6517.000	14007.000	12033.000	14847.000	12894.000

1st week Scores

Tue, Aug 4, 1992 2:29 PM

	Scott Gi	Eric	Chris	N Session Avg.
1				
2				
3	3486.000	0.000	12775.000	6001.625
4	4865.000	10619.000	14490.000	10822.875
5	16114.000	15127.000	15358.000	12664.750
6	16163.000	12271.000	10178.000	11108.125
7	8729.000	15946.000	6475.000	11947.250
8	14924.000	12271.000	16541.000	13004.250

SESSION 2

MODELING SKILLED DECISION-MAKING USING ARTIFICIAL NEURAL NETWORK AND GENETIC-BASED MACHINE LEARNING TECHNIQUES

A THESIS

Presented to

The Academic Faculty

By

Ling Rothrock

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Georgia Institute of Technology

May, 1992

MODELING SKILLED DECISION-MAKING USING

ARTIFICIAL NEURAL NETWORK

AND GENETIC-BASED MACHINE LEARNING TECHNIQUES

APPROVED:

Alexander C. Kirlik, Chairman

Kim J. Vicente

Ronald W. Shonkwiler

Date Approved by Chairperson May 21, 1992

ACKNOWLEDGEMENTS

I would like to thank Professors Laurene and Donald Fausett at Florida Institute of Technology for getting me started in the field of artificial neural networks, and encouraging my continued study in the field.

I would also like to thank the professors at Georgia Tech who have assisted me in my research and in writing this thesis. Professor Alex Kirlik, my advisor, introduced me to the field of human decision-making, and taught me how to model human performance. Professor Ron Shonkwiler took time to critique my artificial neural network models as well as to discuss methods of representing genetic rule strings. Professor Kim Vicente provided valuable feedback on the design chapter of this thesis.

In addition to the professors, I would also like to recognize the assistance of fellow graduate students. Laura Moody was willing to read several thesis chapters and to offer much appreciated suggestions. G. Vidyamurthy was willing to listen to a practice thesis talk and to offer valuable presentation tips. Kelly Deyoe consistently offered me helpful resources on pertinent topics.

Without my graduate research assistantship, I would not have been able to conduct research at Georgia Tech. Therefore, I would like to thank NASA for financial support of my graduate program.

I am grateful to my parents, whose constant encouragement and guidance have helped me through many discouraging times. I am also grateful for Amy Peterson, who is soon to be my wife. Her smile and comforting words have soothed the stresses of writing and rewriting.

Finally, I want to thank God, Who has given me the strength to finish this thesis. Through this thesis, He has also given me a little insight into the wonders of His creation--the human mind.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Computer-based graphical displays are increasingly used as aids to human decision-makers. However, effective use of this new technology requires an understanding of perceptual components of decision skill. As a step toward meeting this need, two sets of models of skilled human decision-making using dynamic, graphically displayed information are presented.

The first model set uses the backpropagation artificial neural network architecture to describe how skilled decision-makers might become increasingly attuned to highly diagnostic features of a graphically displayed decision task. The proposed model is consistent with recent psychological research suggesting that decision-makers develop a "trained eye" through experience. The model set consists of three models which use different input feature representations.

The second model set utilizes genetic-based machine learning techniques to also describe how decision-makers might rely on perceptual components of decision skill. The model set includes a pure genetic algorithm and a revised classifier system. The model sets are similar in that both accept inputs with only dimensional information from the graphical display. The model sets are different mainly in the method used to search the solution space. Whereas the neural networks operate using gradient descent search, genetic-based techniques work by random search.

Results comparing the performance of artificial neural networks and genetic learning models with the performance of human subjects in a laboratory experiment suggest that both model sets provide promising, but different, approaches for the study of decision skill. In particular, the results of one neural network model suggest that relational input features are important cues used by subjects.

CHAPTER I

INTRODUCTION

Current theories of human decision-making reflect a history of retreats from the strong assumptions made by traditional utility theory (von Neumann and Morgenstern, 1947). In utility theory, the worth, or utility, of one's choice is the criteria for decision-making. One who is consistent with the theory would seek to maximize the expected value (i.e. the weighted probabilities of the alternative outcomes of a choice) of one's utility. Although utility theory may be adequate for simplistic scenarios, real-world situations proved too complex to be modeled. A classic example is the Prisoners' Dilemma (Davis, 1985). Two suspects, accused as partners for a crime, are told separately by police that enough evidence exists to warrant X years of jail sentence each. The police further offer the carrot of parole to the suspect who implicates the other. The implicated suspect would have to serve 4X years in prison. However, if both suspects confess, both will have to serve 3X years in prison. The suspects are faced with two options: to deny all charges; or to condemn the partner. The optimal choice for either prisoner is to condemn the other. However, if each condemn the other, the police will be able to implicate both. Therefore, the satisfactory and most stable, though not optimal, strategy is for both to deny all charges. If utility

theory was implemented to solve the Prisoners' Dilemma, each prisoner would always want to maximize his/her utility. Doing so, however, would minimize the utility of both suspects.

To counter utility theory, Simon (1959) proposed, in the context of economics, that the human decision process is oriented to satisfice (to derive a satisfactory solution), rather than to optimize a solution. Characteristics of satisficing behavior are described as follows:

> Models of satisficing behavior are richer than models of maximizing behavior, because they treat not only of equilibrium but of the method of reaching it as well. Psychological studies of the formation and change of aspiration levels support propositions of the following kinds. (a) When performance falls short of the level of aspiration, search behavior (particularly search for new alternatives of action) is induced. (b) At the same time, the level of aspiration begins to adjust itself downward until goals reach levels that are practically attainable. (c) If the two mechanisms just listed operate too slowly to adapt aspirations to performance, emotional behavior--apathy or aggression, for example--will replace rational adaptive behavior. (Simon, 1959, p. 263)

In a complex environment, such as the economic world, the decision-maker is faced with numerous opportunities for action. Thus, information processing limitations constrain the decision-maker to adopt satisfactory solutions. Therefore, models of human decision-making which follow the utility theoretic approach may actually overestimate the ability of humans to process information.

Studies of decision-making in complex tasks confirms Simon's satisficing principle. Lesgold et al. (1988) analyzed X-ray diagnosis by radiologists of varying expertise. They found that diagnoses by expert radiologists were supported by highly refined automatic recognition

capabilities. Novice radiologists, who tended to use a probabilistic approach, were significantly less accurate in their diagnoses. Chase and Simon (1973) examined the ability of chess players of varying skills to generate moves. They found that chess masters were able to extract information from existing board configurations to construct "right" moves for further consideration. Although novice chess players were able to generate the same number of move possibilities as the chess masters, the moves themselves were much less desirable. Thus, in both studies, skilled human decision-making in a complex task did not conform to utility theory prescriptions. In fact, signs of systematic searches through the entire solution space were only evident in novice behavior.

Perceptual Mechanisms of Decision-making

In both the radiologist and the chess player studies, skilled decision-making processes were thought to rely heavily on perceptually-oriented mechanisms that processed displayed information. Lesgold et al. (1988) demonstrated the perceptual processing of experts by their ability to quickly generate a pertinent schema to aid diagnosis. Chase and Simon (1973) noted that it is no mistake of language for a chess master to say that he "sees" the right move. Thus, perceptual components appear to be a major part of expert decision-making in a complex task. To compensate for cases where the perceptual mechanisms cannot uniquely qualify an action, higher cognitive processing may also be required. In light of the importance of perceptual mechanisms in skilled decision-making, three perceptually-oriented modeling approaches are described. The three

approaches are: recognition-primed decision-making in naturalistic environments as proposed by Klein (1989); the research of Hammond and his colleagues on analytical-intuitive cognition (Hammond, Hamm, Grassia, and Pearson, 1987); and the research of Kirlik and his colleagues on perceptual aspects of skilled decision-making in dynamic environments (Kirlik, Miller, and Jagacinski, 1992; Kirlik, Markert, and Shively, 1991).

Recognition-primed Decisions Paradigm

A model of recognitional decision-making in natural settings has been suggested by Klein (1989). The model, recognition-primed decisions (RPD), was formulated through observations and interviews with professionals ranging from fireground commanders to design engineers. The model proposed four major features of decision-making. The first feature involves recognizing an instance as typical. Once an instance has been recognized, the second feature--situational assessment--is activated. Situational assessment involves an understanding of the situation as the decision-maker draws on prior experiences to identify courses of action. Situational assessment concerns not only action goals, but also expectancies to confirm or deny that the correct course has been taken. The third feature, serial evaluation, sequentially assesses options of action until a satisfactory one is found. This feature is most similar to the principle of satisficing. The last feature, progressive deepening, involves mentally pre-playing scenarios to imagine how an option will be executed. Chess

master strategies are examples of progressive deepening. Chess masters tend to chose one move, consider a countermove, and progressively deepen the process (Klein, 1989).

Underlying the RPD model is the assumption that the decision-maker is skilled. Novice decision-makers tend to deliberate concurrently, instead of serially, over available options. The drawback to the RPD model is the lack of formalization. As of now, no successful mathematical implementations of the RPD model have been made.

Intuitive Decision-making Paradigm

Hammond et al. (1987) described an "intuitive" mode of cognition that, at times, out-performs an analytical cognitive mode. In the intuitive mode of decision-making, cues in the environment used by the human are perceptually assessed. The significance of the assessments then determines an appropriate situational judgment. Hammond et al. characterized intuitive decision-making as having the following properties: low cognitive control; rapid rate of data processing; low conscious awareness; and normally distributed errors. Subjects using the intuitive mode of decision-making showed high confidence in the answer and low confidence in the method of deriving the answer.

Through experiments, Hammond et al. found that both task and interface features could influence the type of cognition (analytical or intuitive) used by the subject to perform a task. Thus, a continuum, ranging from purely analytical to purely intuitive, was devised to account for the possible strategies for performing a task.

Unlike RPD and perceptual decision-making paradigms, intuitive decision-making involves a "natural" mode of perceptually-oriented processing. A natural mode denotes that the processing did not depend on training at the task.

Perceptual Decision-making Paradigm

Kirlik, Markert, and Shively (1990) suggested that the human perceptual system serves a dual role in decision-making performance. First, the perceptual system functions as a mediator between the context of a current decision process and stored knowledge. Second, the perceptual system can directly initiate activity through the development of a "trained eye." Kirlik (1992) characterized a perceptual model of skilled decision-making in a dynamic task. His model was based on a hierarchically structured set of perceptual mechanisms that defined a set of decision options. The mechanisms also defined a distribution of values over the options to indicate the availability of each.

Kirlik, Miller, and Jagacinski (1992) conducted empirical studies to identify the perceptual mechanisms used by human subjects while interacting with a computer simulation of an aviation micro-world. The micro-world consisted of five friendly aircrafts, a computer-generated terrain, enemy craft, and cargo. Friendly aircraft "crews" were chosen from a university population. The objective was to pilot a scout aircraft through the terrain and dispatch the other four aircrafts to engage enemy targets or load and unload cargo. A task analysis of the human-environment system led to the design of a process model capable of

mimicking subject behavior. The process model categorized displayed situations in terms of the degree to which various actions were appropriate. The categorization was mainly accomplished by perceptual components. As the micro-world was a dynamic environment, immediate perceptual classification was necessary. The effectiveness of the process model was shown by the similarity of actions suggested by the model's perceptual mechanisms and those taken by the crews.

The approach of investigating the perceptual mode of decision-making has been the most formal of the three paradigms described (RPD, Intuitive, and Perceptual). Thus, the perceptual approach seems best suited to be modelled through mathematical techniques.

Graphical Considerations for Experimental Design

The design of graphical display for a task to analyze perceptual decision-making processes must be carefully considered. Barnett and Wickens (1988) proposed the principle of compatibility of proximity to aid display design. The principle asserts that multiple channels of information requiring mental integration should be physically integrated as well. In support of the principle, they showed that integration of information using rectangles was significantly better than using bar graphs. Even when the task required focussing attention on isolated attributes, subject performance using the integrated rectangle display did not suffer. Barnett and Wickens also noted that an important element in the principle is the

emergent feature. An emergent feature is a property of the configuration of multiple dimensions of an object that does not exist when the dimensions are specified independent of one another.

Sanderson, Flach, Buttigieg, and Casey (1989) also stressed the importance of emergent features in aiding display design. The emergent feature principle was used to predict the effectiveness of visual displays in supporting a variety of monitoring tasks. Emergent properties of a visual stimulus should identify invariants in the environment. An invariant is a permanent physical property of an object or system that remains constant across all superficial transformations unless an abnormality is present (Sanderson et. al., 1989).

Modeling Goals and Objectives

The purpose of this research was to construct mathematical models of the perceptual decision-making paradigm using artificial neural network and genetic learning methodologies. To analyze human perceptual decision-making processes, subjects' performances on a dynamic decision-making task were recorded. The mathematical models were then used to simulate subject performance on the same task. By analyzing the performance of the mathematical models, certain diagnostic features could be inferred. In particular, relational input features were found to significantly improve the performance of an artificial neural network model. To further examine the neural network model, a "skeletonization" technique was used to identify how the trained model had become attuned to various input features.

Chapter 2 describes the dynamic graphical task used to investigate skilled decision-making. It also illustrates a method to generate a profile of errors, called "error signature diagnosis", in order to identify perceptual heuristics from behavioral data. Chapter 3 describes artificial neural network models of subject performance in the graphical decision task. A brief introduction to artificial neural networks is given first. Next, a neural network model is presented in terms of the graphical decision task and components of perceptual decision-making. A specifications section follows to discuss implementation of the neural network in terms of the graphical decision task and perceptual decision-making components. Chapter 4 describes a genetic algorithm and a genetic-based machine learning model of subject performance in the graphical decision task. Traditional genetic algorithms are first introduced and explained. Revisions of the traditional structure to accommodate the graphical decision task are also described. In addition, a genetic-based machine learning model is described in terms of the graphical task and components of perceptual decision-making. Chapter 6 discusses computer program design issues. Chapter 7 presents the analysis of results generated by the artificial neural network, genetic algorithm, and genetic-based machine learning models. The skeletonization technique is also described there. Chapter 8 concludes the thesis with a discussion of significant findings and implications for further research.

CHAPTER II

TASK AND EXPERIMENTS

Laboratory Task

In 1980, Tulga and Sheridan constructed a scheduling task to explain that human decision-making was based on principles of utility theory. Kirlik, Markert, and Shively wanted to explore ramifications of applying simpler perceptual models to the same scheduling task. The task was modified to promote perceptual mechanisms of skilled decision-making. The modified task will be referred as the revised Tulga task (RTT). A description of the task is as follows:

> The display [see figure 2-1] shows four horizontal lines. At the start of each trial, four rectangles of variable height and width appear at the left side of the screen and begin to move to the right. Each rectangle represents a task that must be processed before it reaches its due-date (the end of the line). In the present version of the task, subjects processed a rectangle by pressing a numbered key. The keys were vertically arranged on a modified keyboard to ensure spatial compatibility with the display. In general, the subject's task is to determine an appropriate order in which to process the four rectangles and to enter the order as rapidly as possible.
>
> Only one rectangle can be processed at a time. If the subject enters the keys 1, 2, 3, 4, in that order, rectangle 1 will begin processing and the other rectangles will travel some variable distance across the screen during the time in which rectangle 1 is being processed. After the first rectangle has completed processing, the next rectangle will begin processing, and so on, until all the rectangles have disappeared from the screen. The subject cannot interrupt the ongoing processing of a rectangle, and may make keypresses at any time after the initiation of the trial, although a performance penalty is incurred if the subject delays between keypresses. As an inducement to rapidly determine an ordering of the rectangles, any time delay incurred between

entering successive keys is reflected in a delay of the same duration between the end of the first rectangle's processing and the start of the second rectangle's processing. This feature of the task was included specifically to promote the development of rapid, possibly perceptual decision processes.

During intervals in which a rectangle is being processed, or in which processing is delayed, the remaining rectangles move across the screen at variable speeds. The speed with which a rectangle moves has been constrained to vary linearly with its area. Larger (area) rectangles move proportionately slower than smaller rectangles.

The optimal ordering is a function of a number of factors. First, the nominal value of a rectangle is indicated by its height. In calculating the actual payoff earned by processing a rectangle, though, the nominal value is divided by the distance the rectangle has traveled when it has completed processing. Therefore, a higher proportion of a rectangle's nominal value is earned the shorter the distance it has traveled. The overall payoff for a single trial is the sum of the payoffs earned for the four rectangles. (Kirlik, Markert, and Shively, 1990, p. 519)

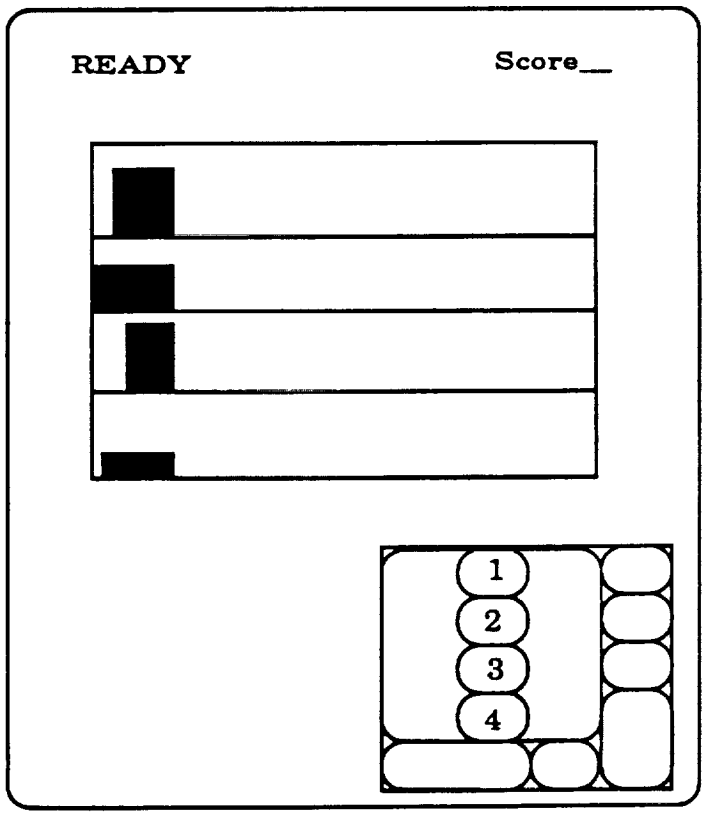


Figure 2-1. Revised Tulga-task Display

The optimal payoff score is calculated as follows:

Given the 24 possible orderings for the four rectangles,

optimal_order_payoff = max total_payoffs(i) for i=1,..,24

where the total payoff score is the summation of partial payoff scores of each rectangle selected at the jth order

total_payoffs(i) = Σ partial_payoff(j) for j=1,..,4

and

$$\text{partial\_payoff}(j) = H(j)^2 * W(j) / \Sigma W(k) \text{ for } k=1,..,j \qquad (2.1)$$

where H is the height and W is the width of RTT rectangles.

Subjects receive a score at the end of each trial, indicating the percent of optimal performance that was achieved. The subject payoff score, which range [0, 100], is calculated as follows:

payoff = Σ actual_payoff(l) / maximum_payoff for l=1,...4.

where actual payoff is the score at each rectangle selected at lth order by the subject, and

$$\text{actual\_payoff}(l) = H(l)^2 * W(l) * C1 / (\Sigma [C2 * W(m)] + \Sigma [C3 * \text{delay\_time}(m)]) \qquad (2.2)$$

for m=1,..,l. C1, C2, and C3 are constants and delay_time(m) is the time between subject order selections.

Experiment

Four subjects (1, 2, 3, 4) from the student population of Georgia Tech participated in the experiment. Students received monetary compensation for each hour of participation. The participants were encouraged to correctly order the rectangles so as to achieve the highest possible payoff

score. The individual with the highest average payoff score at the end of the experiment was given a monetary bonus. The subjects were told that the rectangle height represented the nominal payoff value. They were also told to order the rectangles as quickly as possible.

Each experimental session consisted of a series of 80 trials. At the beginning of each trial, a 5 sec. pause was given. Following the pause, a READY message, as shown figure 2-1, was shown for 3 sec. The rectangles then appeared and started moving to the right at varying speeds (according to area). After all the rectangles were processed, the payoff score was displayed for 2 sec. Following the payoff score display, the 5 sec. pause was repeated and the process continued until the completion of 40 trials. At the end of 40 trials, the subjects were instructed to take a short break, after which the remaining 40 trials were completed. Each session typically lasted one hour.

Two Macintosh II's were used to run the experiment. The keypad was modified to allow rectangle ordering. As shown in figure 2-1, the "1" key represented the top rectangle, the "2" key corresponded to the second from the top, the "3" key meant the third from the top, and the "4" key represented the bottom rectangle. A set of 16 rectangles were used to generate different combinations of four rectangles to appear on the computer screen. A graphical representation of the 16 rectangles in ln(height) X ln(width) space is shown is figure 2-2.

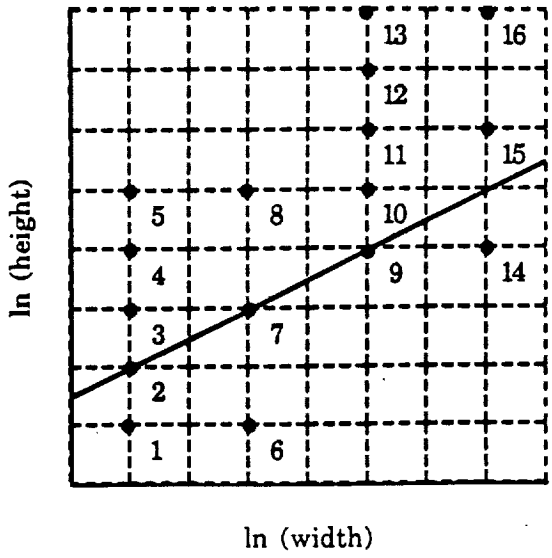


Figure 2-2. Rectangles Used in the Experiment

The subjects completed a total of five sessions over a five day period. Each day's session was the same for all four subjects. Day 1 was used as a practice session. Therefore, only data from days 2 to 5 were used for modeling purposes. Data captured from the sessions include reaction time (delay time between keypresses), order of rectangles, and payoff score.

Task Analysis

From the description of how optimal payoff scores were calculated, a pair-wise rectangle ordering rule can be generated. Given two rectangles **a** and **b**, rectangle **a** should be processed prior to rectangle **b** if (from *(2.1)*):

$$H_a^2 + H_b^2 [\, W_b / (W_a+W_b) \,] \geq H_b^2 + H_a^2 [\, W_a / (W_a+W_b) \,]$$

Isolating features of each rectangle, the inequality then becomes:

$$H_a^2 / W_a \geq H_b^2 / W_b \qquad (2.3)$$

To graphically represent the optimal pair-wise ordering rule *(2.3)*, make H_a and W_a dependent variables (H and W respectively) and H_b and W_b

independent variables. Furthermore, simplify *(2.3)* to an equality to depict a line instead of a region. Thus, the rule becomes:

$$H^2 / W = H_b^2 / W_b$$

Assuming $W > 0$ and $W_b > 0$, take logarithms of both sides:

$$\ln(H^2 / W) = \ln(H_b^2 / W_b)$$

We get:

$$2 \ln H - \ln W = 2 \ln H_b - \ln W_b$$

Letting $y = \ln H$ $\quad y_1 = \ln H_b$
$x = \ln W$ $\quad x_1 = \ln W_b$

and substituting, we get:

$$(y - y_1) = 1/2 (x - x_1)$$

which is the equation of a line in Cartesian coordinates. Thus, the pair-wise ordering rule is represented by the slope of the line in ln(height) X ln(width) space. Given a slope m, and rewriting the equation of a line to a more familiar form, we get:

$$y = mx + b \quad \text{where} \quad b = -mx_b + y_b$$

In figure 2-2, the line through rectangle 7 shows its ordinal position among all the rectangles on the grid. Thus, each rectangle satisfying:

$$y > 1/2\, x + b$$

will be pair-wise ordered prior to rectangle 7. Similarly, each rectangle satisfying:

$$y < 1/2\, x + b$$

will be pair-wise ordered after rectangle 7.

Kirlik, Markert, and Shively (1990) devised a method of calculating "error signatures" to identify possible perceptual heuristics that could be used to pair-wise order the rectangles. The error signature uses the line representing the optimal heuristic *(2.3)* to compare with lines generated by other rules. To calculate the error signature, bound the deviations of the non-optimal rule from the optimal rule. Consider m as the slope of the heuristic to be compared, one set of constraints is as follows:

$y > 1/2\,x + b$ $\quad$ $y > mx + c$ $\quad$ where b and c are constants.

The second set of constraints is similar:

$y < 1/2\,x + b$ $\quad$ $y < mx + c$

The intersection created by the two sets of constraints represents the error region. For instance, given the rule:

$$H / W > H_b / W_b \qquad (2.4)$$

which has a slope of one, the intersection created by the optimal rule and the non-optimal rule contains rectangle 15. Thus, if a subject consistently mis-orders the pair of rectangles 15 and 7, the error signature profile can show that he/she is using rule *(2.4)*. Therefore, the error profile can show if subject behavior was consistent with a particular rule.

CHAPTER III

ARTIFICIAL NEURAL NETWORK MODEL OF PERCEPTUAL DECISION-MAKING

Introduction to Artificial Neural Networks

To better understand components of decision skill, a model of human decision-making using dynamic, graphically displayed information was constructed. Using the RTT as the graphical task, the model sought to simulate performance as indicated by the data of the four human subjects. The model utilized the artificial neural network (ANN) paradigm. ANNs are biologically-inspired systems that process information via dense interconnections of simple computational units. ANNs will be introduced in the present section with respect to the decision-making paradigm. A more technical explanation will be included later. ANNs consist of two primary elements: computational units (or nodes) and interconnections (or weights). Figure 3-1 shows an ANN with two layers of weights and three layers of nodes. The nodes receive input signals from other nodes via the weights. Based on the strength of incoming signals, the receiving node, in turn, generate an output signal. The strength of the output signal is determined by an activation function residing on the receiving node. Given a target criterion, an ANN can learn by using training rules to change weights.

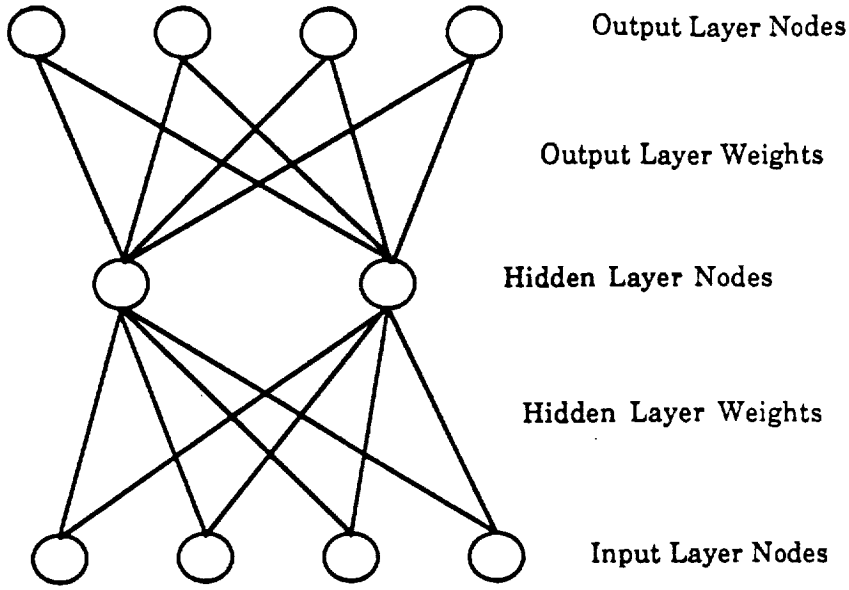


Figure 3-1. Artificial Neural Network Architecture

Artificial Neural Network Characteristics

Parallel-processing ANNs have several benefits over traditional von Neumann sequential computers (Lippmann, 1987). Due to massive interconnections among processing nodes, ANNs typically provide a greater degree of robustness or fault tolerance. In addition, most ANN have the ability to adapt and learn over time. Moreover, ANNs make weaker assumptions regarding the solution space. ANNs "Don't 'execute programs' as much as they 'behave' given a specific input ... They 'react,' 'self-organize', 'learn,' and 'forget'" (Caudill, 1987, p. 48). Thus, ANNs make promising models of human performance for certain types of tasks.

There are basically four types of ANN's (Fausett, 1992): an associative memory, which is trained to associate a set of input vectors with a corresponding set of output vectors; a pattern classifier, which classifies

an input vector as belonging or not belonging to a category; a self-organizer, which groups similar input vectors without external supervision; and a constrained optimizer, which solves constrained optimization problems.

Neural Network Models of Cognition

Three issues must be addressed in order to construct a neural network model of human cognitive processes. The first issue concerns the structure of the network. Rumelhart, Hinton, and McClelland (1986) described three types of structure: bottom-up, top-down, and interactive. The fundamental characteristic of the bottom-up structure is that lower level elements can only communicate with upper level ones. In the top-down structure, on the other hand, only upper level elements can communicate with lower level ones. With interactive structure, two-way communication is possible between levels. The second issue pertains to the function of neural network system components. Rumelhart, Smolensky, McClelland, and Hinton (1986) described a neural network which simulated a person's mental model of a situation as having two sub-networks with two distinct functions. One sub-network interprets the inputs from the environment, while the other sub-network produces expected outcomes based on the interpretations. The third issue focuses on the types of input representation used by the network. Humans have three essential abilities that allow them to come to logical conclusions without being logical (Rumelhart, Smolensky, McClelland, and Hinton, 1986, p. 44): they are good at pattern matching; they are good modelers of the world; and they are good at manipulating the environment around them. Therefore, to

effectively simulate the task environment, the network designer must realize that artifacts--man-made physical representations created to simplify problem solving--exist, and are used extensively as perceptual inputs to the human cognitive system.

A brief survey of three existing psychological neural network models is presented. The first model simulates human skill acquisition by using several connected ANNs. The second is a configural network model of human classification and recognition learning using a one-layer ANN. The third is a decision-making model utilizing three connected ANNs.

The first model, proposed by Schneider and Detweiler (1988), was able to simulate skill acquisition in a word recognition task. The model consists of three levels--microlevel, macrolevel, and system-level. The microlevel represents a network of neuron-like units that process input information from the environment. The inputs to the microlevel are represented as binary strings, where 1 or 0 denote the presence or absence of a feature. Microlevel units are organized into functional modules. Modules categorize sensory (i.e. auditory, visual, motor) and language (i.e. semantic, speech, context) information available to the network, and can be effected by the network. The macrolevel, then, represents interactions among the modules. Macrolevel modules are further organized into regions of sensory-language interaction. Finally, the interactions are interpreted at a system-level. At the system-level, a central control structure receives reports from all regions and regulates the transmission of instructions back to the modules.

Schneider and Detweiler were able to simulate phases of human skill acquisition through the model. Model behavior in the initial trials resembled controlled processing where performance was slow and effortful. After a period of training, the model was able to perform using "automatic" processing (i.e. direct associative retrieval of output patterns from input patterns).

The second model, proposed by Gluck, Bower, and Hee (1989), was able to account for learning results from several animal and human learning literatures. The model consists of a one-layer pattern classifier ANN with "configural" input features. A configural model encodes pair-wise conjunctions of stimulus features as unique elements. Non-linearly separable problems can be overcome by explicitly coding "higher-order" elements through conjunctions of elementary features. The inputs are coded in binary where 1 or 0 denotes the presence or absence of a configural feature.

Gluck et al. tested the configural model on three tasks. The first task involved solving non-linearly-separable classifications. The second task entailed judging test patterns as "old" training instances or new instances not experienced before. The third task consisted of classifying "noisy" data (i.e. previously trained instances with characteristics added or missing) into categories. In all tasks, the performance of the configural model was found to be similar to that of the human subject. Gluck et al. attributed the

success of the configural model to an implicit exponential decay relationship between input cues. The same relationship exists between stimulus similarity and psychological distance.

The third model, proposed by Leven and Elsberry (1990), attempted to simulate human skill-acquisition and problem solving using a system of ANNs. The model asserted that humans learn and analyze through three hierarchical methods. First, humans perform in a calculative, rational manner through the use of context-free rules. The first method represents the least skilled phase of human learning, and is modeled as an associative memory ANN. Second, humans use a structured, analytic mode of learning via Bayesian-processing. The second method depicts a training phase, and is represented as a pattern classifier ANN. Third, humans process context-sensitive information through "intuitive" guidance. The third method reflects a skilled phase, and is represented as a self-organizing ANN.

Performance of Leven and Elsberry's model resembled characteristics of Simon's satisficing principle. As performance of the ANN models fell below a level of aspiration, rules were reconstructed in the data base to change search behavior. However, if the performance indicated that the goal cannot be attained, model behavior became radical and unpredictable.

Psychological neural network models mentioned in this section have been successful in simulating performance for their respective tasks. For the RTT, however, the same models would need to be drastically altered to

simulate human performance. Thus, a different strategy was needed to model human decision-making in the RTT. The strategy chosen was the perceptual decision-making paradigm. As perceptual decision-making relies on a pattern recognitional component, a pattern classifier ANN was selected as the model for the RTT.

Constraints on Artificial Neural Network Model of Decision-making

Kirlik and Rothrock (1991) suggested several constraints for the ANN model of the RTT. First, input features must be determined by close examination of perceptually available displayed information. In the RTT, the physical dimensions of the rectangles are assumed to be the most pertinent perceptual information. Second, artifactual input features (dimensions not intentionally designed to communicate meaningful information) must be considered. Shape and area, for instance, should not be neglected. Third, input feature salience, as opposed to diagnosticity, should not be neglected. For the ANN model of the RTT, feature salience is assumed to also be the feature diagnosticity. Last, the attended feature set, which may be a small subset of all information available in the environment, should be extracted. Although some input features can be ignored by the ANN, additional features cannot be added. Therefore, the selection process must be thorough.

Backpropagation Algorithm

The pattern classifier ANN chosen to model the RTT used the backpropagation algorithm and a three-layer architecture. The Generalized Delta Rule (Rumelhart, Hinton, and Williams, 1986), or

backpropagation algorithm, contains two phases: feedforward of input signals and backpropagation of errors. Given the same architecture shown in figure 3-1, the feedforward phase consists of the following: each input layer node sends its input signal, multiplied by connecting hidden layer weights, forward to hidden layer nodes; each hidden layer node then sum all inputs and applies its activation function to determine its output signal; the output signals, multiplied by connecting output layer weights, becomes input signals to output layer nodes; finally, each output layer node sums all inputs and applies its activation function to determine the outputs of the network. The backpropagation phase consists of the following: the outputs of the network from the feedforward phase are compared with a set of target nodes; the errors (target-output) are propagated back through the network via the learning rule; finally, the hidden and output layer weights are updated based on the propagated errors.

Backpropagation Model of Decision-Making in RTT Context

In terms of the RTT, the input signals to the backpropagation network were representations of the rectangles. The outputs of the network represented an ordering of the rectangles. To factor out motor skill effects, the reaction times of the network were made to match subject reaction times for the same session. In addition, the network was trained on the same trials as the subjects.

The backpropagation algorithm contains some strengths which make it useful for modeling perceptual decision making. First, cue attunement can be thought of as network learning. As the network learns

information through its weights certain input nodes become more diagnostic than others. Second, once high diagnosticity is achieved, i.e. the network is trained, the feedforward component alone can be used to simulate a mode of decision-making where perception directly specifies action.

There are weaknesses to the modeling paradigm as well. First, an underlying assumption of the backpropagation algorithm is that all relevant input features can be identified by the modeler. Thus, the network cannot add supplementary cues from outside the current representation. Second, the backpropagation network tends to require thousands of training epochs (cycles of input signals). The subject, however, was trained only once on the same set of input signals. Third, the target representation of optimal orders is not available in the RTT. The only feedback received by the subject was the payoff score. Fourth, training rules (search techniques) used by the backpropagation algorithm, such as gradient descent, sometimes cannot overcome local minima problems. Whereas the human subject could select a different diagnostic cue from the display, the ANN is trapped in a fixed search scheme.

Artificial Neural Network Specifications

The present section describes the implementation of the backpropagation algorithm in the context of the RTT. A primary modeling objective was consistency and generality. Three different network representations were considered. To objectively evaluate each network's ability to learn and adapt, the following set of criteria was used: the initial

weights, created by a pseudo-random number generator, were the same for all three ANNs; the number of hidden layer nodes was n/2 where n was the number of input layer nodes; the learning rule was the Least Mean Square (LMS) rule; the activation function was the logistic sigmoid function for inputs ranging from -1 to 1; values for input and output layer nodes were bound between 0 and 1; and the threshold for the error term was set at 0.165.

To facilitate robust learning, weights were created to range from -1 to 1. The number of hidden layer nodes was arbitrarily chosen. Caudill (1988) noted that too many hidden nodes will encourage the network to memorize patterns, while too few will drastically extend the number of training iterations. The LMS rule essentially reduces the difference (error) between the target output and the actual output generated by the network through gradient descent. The error of each pattern **E** is:

$$E = \frac{1}{2} \sum_{m=0}^{k} (d_m - y_m)^2$$

where **k** is the number of output layer nodes, d_m is the target node, and y_m is the output layer node. Fausett (1992) suggested using the bipolar form of the logistic sigmoid function **f(x)**, and the corresponding derivative **f'(x)** for binary input data where:

$$f(x) = \frac{2}{1 + \exp(-x)} - 1 \qquad \text{and} \qquad f'(x) = 0.5\,[1 - f^2(x)]$$

Figure 3-2 shows a simplified version of the feedforward phase of the backpropagation algorithm for the RTT. For the actual RTT model, the

number of input and hidden layer nodes was determined by the input representation, and the number of output nodes remained four. The input layer nodes (**0** and **1**) fed forward signals from outside the network. The signals were multiplied by weights and summed as $\mathbf{S_{2,3}}$ at the hidden layer nodes (**2** and **3**). The activation function, which was the same for all nodes, then calculated the output of the hidden layer nodes $\mathbf{y_{2,3}}$. These outputs, multiplied by weights, were then summed as $\mathbf{S_{4,5}}$ at the output layer nodes (**4** and **5**). Finally, the activation function at the output layer nodes calculates the output (i.e. ordering of the rectangles) of the network $\mathbf{y_{4,5}}$.

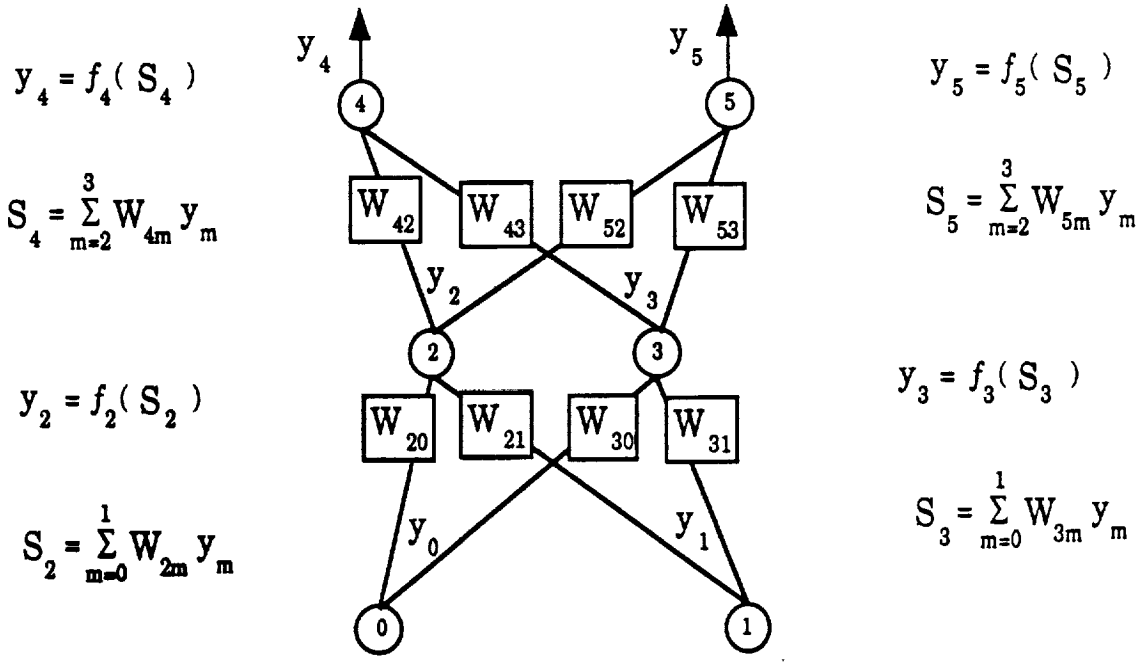


Figure 3-2. Feedforward Phase of Backpropagation Algorithm

Figure 3-3 shows a simplified version of the backward propagation phase of the backpropagation algorithm for the RTT. μ represents the learning rate, which decreases with time. Correction terms ($\partial_{4,5}$) on the

output layer nodes were calculated based on the error of the target ($\mathbf{d_{4,5}}$) and output ($\mathbf{y_{4,5}}$) nodes. The terms were then propagated back the network so that hidden layer correction terms ($\partial_{2,3}$) could be calculated. Finally, the weight corrections ($\Delta\mathbf{W}$) were used to update weights for the current input signals. The new weights were calculated as:

$$W_{ij}^{(new)} = W_{ij}^{(old)} + \Delta W_{ij}$$

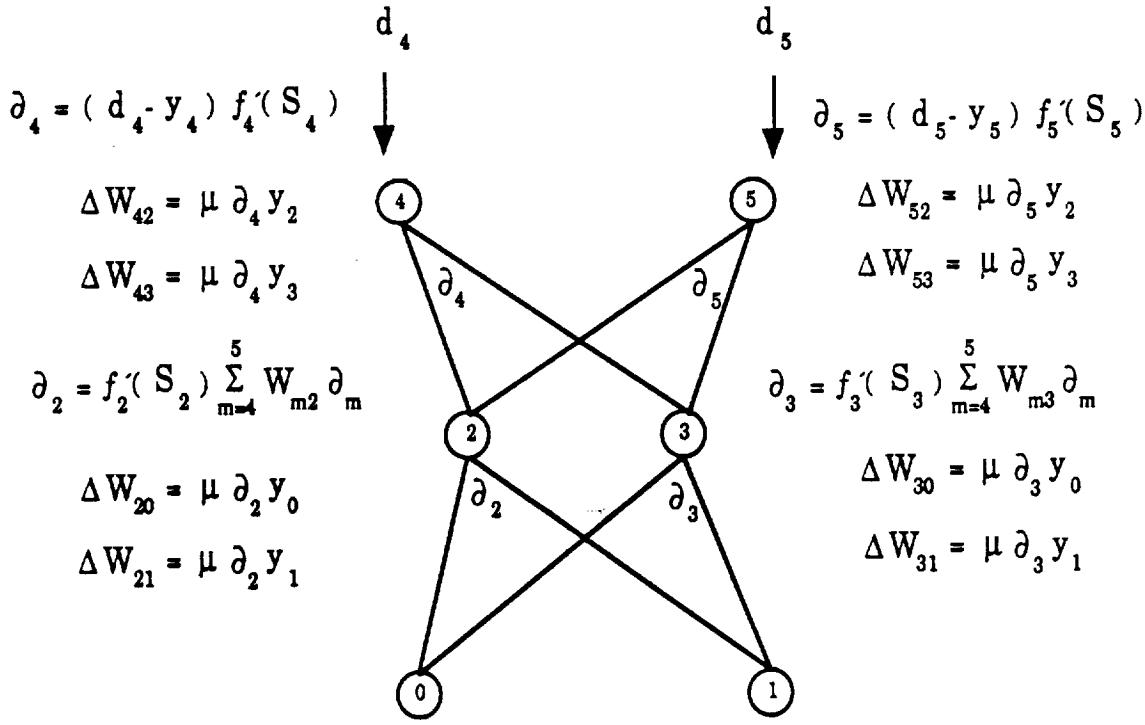


Figure 3-3. Backward Propagation Phase of Backpropagation Algorithm

Input Representations

Three input representations for the RTT are shown in figure 3-4. The dimensional representation contained scaled (from 0 to 1) real number values of the height (H), width (W), area (H * W), shape (H ÷ W), and the

optimal heuristic ($H^2 + W$) (Kirlik, Markert, and Shively, 1990) of each rectangle. In terms of the signature error discussed in Chapter II, each dimension (which can also be considered a heuristic for the subject) can be represented as a line in ln(height) X ln(width) space. The slope of the height line is 0, the width line is undefined, the area line is -1, the shape line is 1, and the optimal heuristic is 1/2. There were 20 input layer nodes for the dimensional representation. The same dimensions were considered in the relational representation. However, the value of the relational nodes were binary, where 1 represented that the rectangle had the greatest dimensional value of all the rectangles in the present input set. There were 20 input layer nodes for the relational representation. The third representation applied the strategy of the relational representation to all possible pairs of rectangles. Thus, the rectangles with the greater dimensional value in each pair were assigned a 1. There were 30 input layer nodes for the pair-wise relational representation. This representation was similar to the configural representation used by Gluck, Bower and Hee.

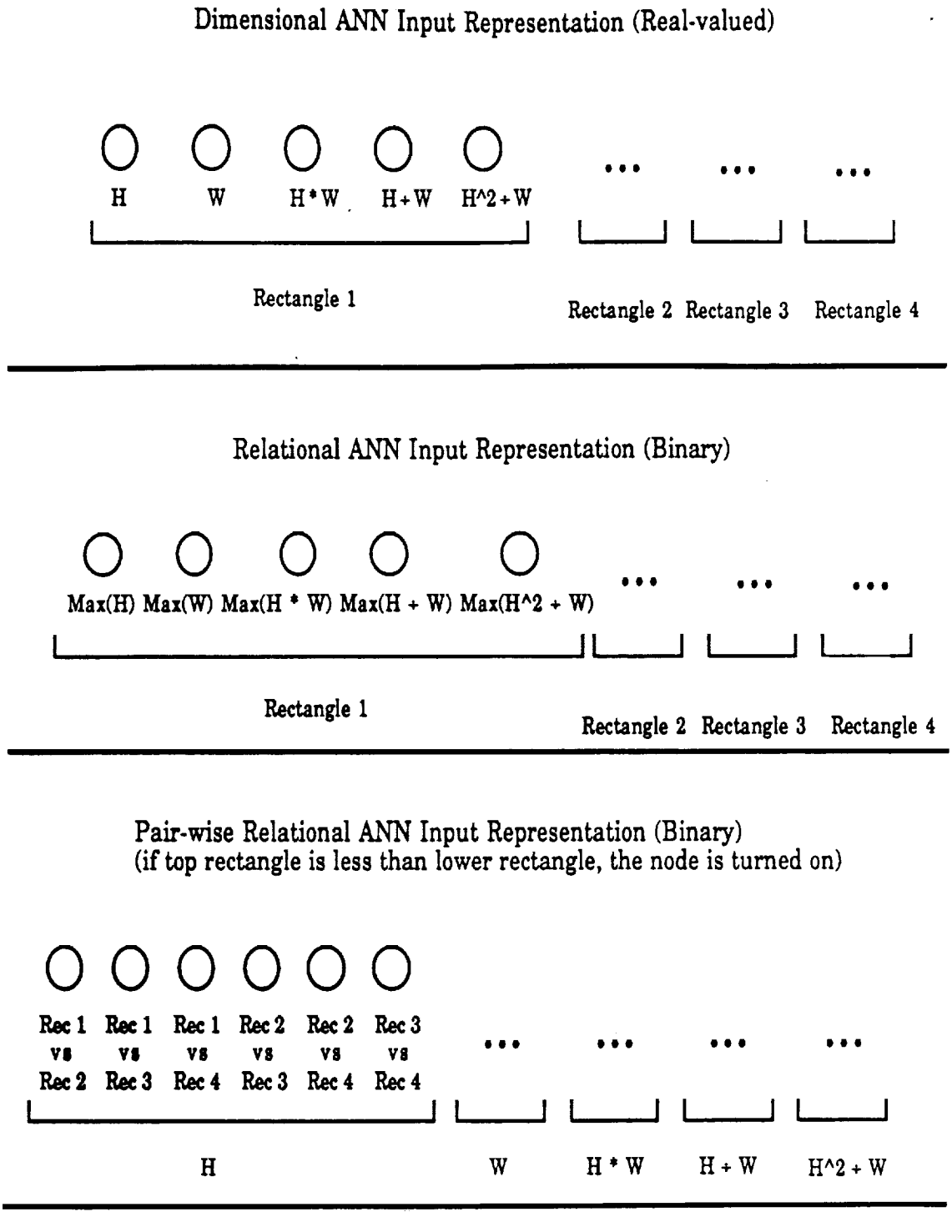


Figure 3-4. Artificial Neural Network Input Representations

RTT Modifications to the Backpropagation Algorithm

Figure 3-5 shows the flow of the backpropagation model of the RTT. Some modifications were made to the feedforward of data and back propagation of error phases. From the input layer nodes to the output layer nodes, conventional feedforward processing took place. The output layer nodes, which were real-valued, represented strength. To convert the strength to a rectangle order, the output layer nodes were first assigned rectangle identifiers. The first output node, for instance, would be assigned the top rectangle (in relation to other rectangles as it appears on the screen) in the RTT. The second output node would be assigned the second rectangle from the top, and so on. Thus, if the strength of the third output layer node was the strongest, the third rectangle from the top would be selected first in the order.

Once the rectangle order was established, the RTT payoff score was calculated. The computation of the payoff score also yielded the optimal order for the present rectangle set. Using the optimal order, or subject order if training on subject data is desired, the values of the target nodes were set by reversing the procedure of converting strength to order. Converting the cardinal order to a strength value, the first was assigned a 1.000000 value, the second was assigned a 0.666666, the third was assigned a 0.333333, and the fourth was assigned a 0.000000. The target values were set to provide maximum separation within the bounds of [0,1].

Once the target node values were set, conventional back propagation of errors occurred and the weights were adjusted. After the weights were

updated, the next input signal pattern repeated the process. Training for the backpropagation model continued until either a predetermined number of epochs (1 epoch represents a cycle of all input signal patterns) had been reached, or learning was complete.

Learning was deemed complete if a certain threshold criteria was reached. The error (i.e. target-output) was compared to a fixed threshold of 0.165. If a significant amount of error surpassed the threshold, learning was not complete. Thus, the range of allowable value for nodes corresponding to rectangle strength was as follows:

0.000000 -> 0.165000 for the rectangle ordered last

0.168333 -> 0.498333 for the rectangle ordered third

0.501666 -> 0.831666 for the rectangle ordered second

0.835000 -> 1.000000 for the rectangle ordered first

The threshold criteria was designed to constrain outputs to numerical boundaries. If the error was within the allowable range, the ordering would be optimal.

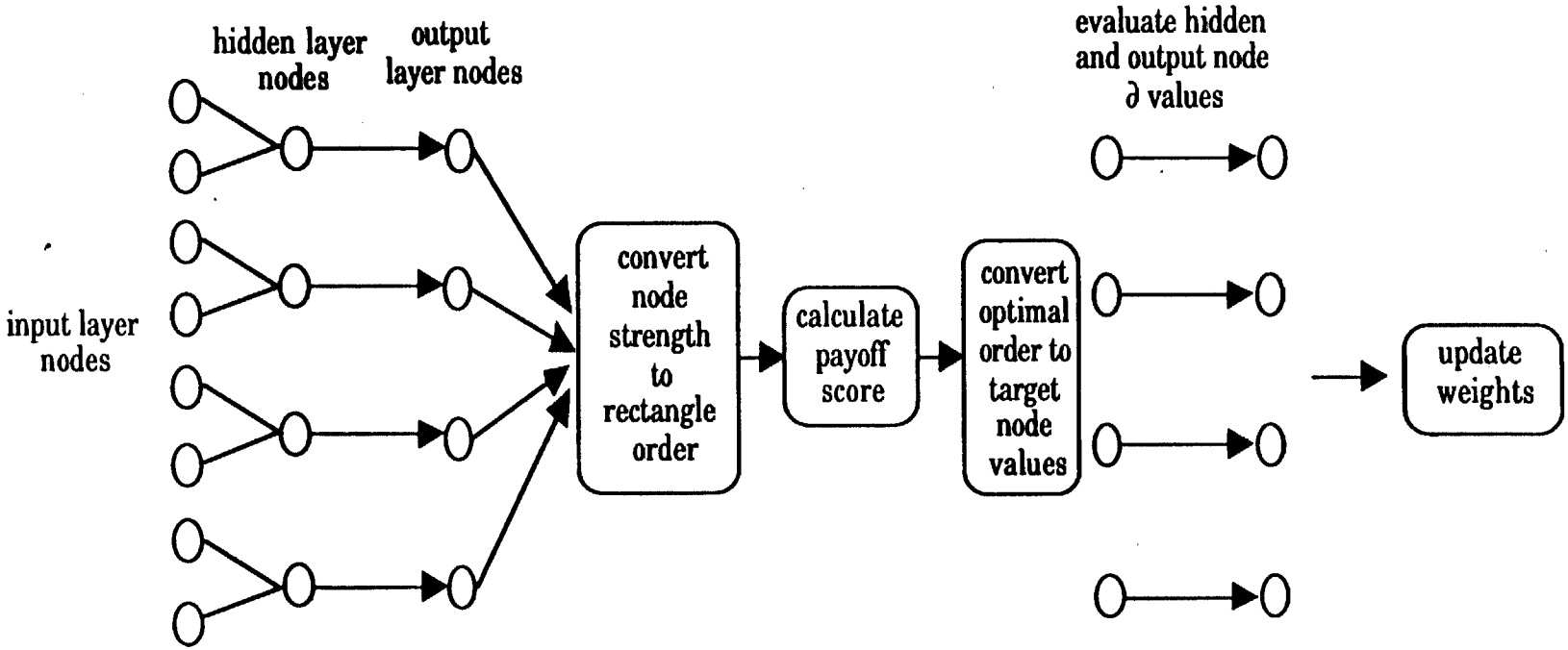


Figure 3-5. Flow of Backpropagation Model

Summary

This chapter attempted a comprehensive treatment of ANN-inspired human decision-making models. An brief introduction to ANNs was first given to acquaint the reader with the paradigm. A literature search of existing ANN models of cognition revealed the need for a perceptually-oriented decision-making model. Under constraints of the perceptual decision-making methodology, a rationale supporting a backpropagation model of the RTT was developed. Finally, details describing the various adaptations of the backpropagation model to the RTT were discussed.

CHAPTER IV

GENETIC MODELS OF SKILLED DECISION-MAKING

Introduction to Genetic Algorithms

Genetic algorithms (GAs) are search algorithms based on the theory of biological evolution and principles of natural genetics. In computational terms, Grefenstette (1990) suggested that GAs are distinguished from other search techniques by the following features: A *population of structures* that can be interpreted as candidate solutions to the given problem; the *competitive selection* of structures for reproduction, based on each structure's fitness as a solution to the given problem; and idealized *genetic operators* that recombine the selected structures to create new structures for further testing. The power of GAs lies in the ability to adapt to combinatorially explosive search spaces about which little can be known a priori (De Jong, 1990).

GAs have several benefits over traditional optimization and search techniques (Goldberg, 1989). GAs make no assumptions regarding the problem space. The "blind" search is solely driven by use of objective function information. Also, instead of searching a single point in the solution space, GAs search in parallel from a population of points. In addition, GAs work with the coding of a parameter set, not the parameters themselves.

GA models are able to overcome some drawbacks of ANN models. First, GA models are able to code a parameter set of input features to create different combinations, whereas the ANN model's input features are fixed. Second, the ANN model's local minima problem is not encountered by GA models because of probabilistic search. The final difference concerns the RTT in particular. The GA model's objective function is a scalar measure of the fitness of a GA structure; similar to the payoff score function in the RTT. The criterion for the backpropagation algorithm, however, is a vector measure of correctness in the ordering of rectangles.

Genetic Algorithm Characteristics

The unit of analysis in a GA model is the gene. Just as human genes carry bits of biological data, GA genes carry bits of data about an object of analysis. The object could be anything from a solution for an undefined function to a component for a computer program. For humans, the genes combine so that bits of data become a stream of information known as the chromosome. The GA analogy to the chromosome is the string. A collection of chromosomes, or strings, is known as a population. In most GA models, the number of strings within a population remains constant.

A GA model can operate on genetic strings using three operators: reproduction, crossover, and mutation. A sample model is shown in figure 4-1. G_{ij} represent the jth genetic data contained in the ith string. The initial string population is homogeneous because no interaction between strings has taken place. The first GA operator, reproduction, can replicate an existing string based on the string's fitness. A string is deemed "fit" if it

evaluates to a high objective function value. In the sample model, the second string was reproduced. The second GA operator, crossover, can exchange information between strings. Crossover randomly selects a site at which two strings are "spliced." Shown as phase II in the sample model, the first and second strings were spliced between genes 4 and 5. Through splicing, the variation within the population increases. The third GA operator, mutation, randomly changes genetic data in a string. Sometimes, reproduction and crossover may indiscriminantly destroy useful genetic material. Therefore, the mutation operator is needed to protect against irrecoverable loss by bringing new information into the population. Mutation is shown as phase III in the sample model. $G_{2,5}$ in the third string was randomly replaced with M. The new population represents the next generation of string.

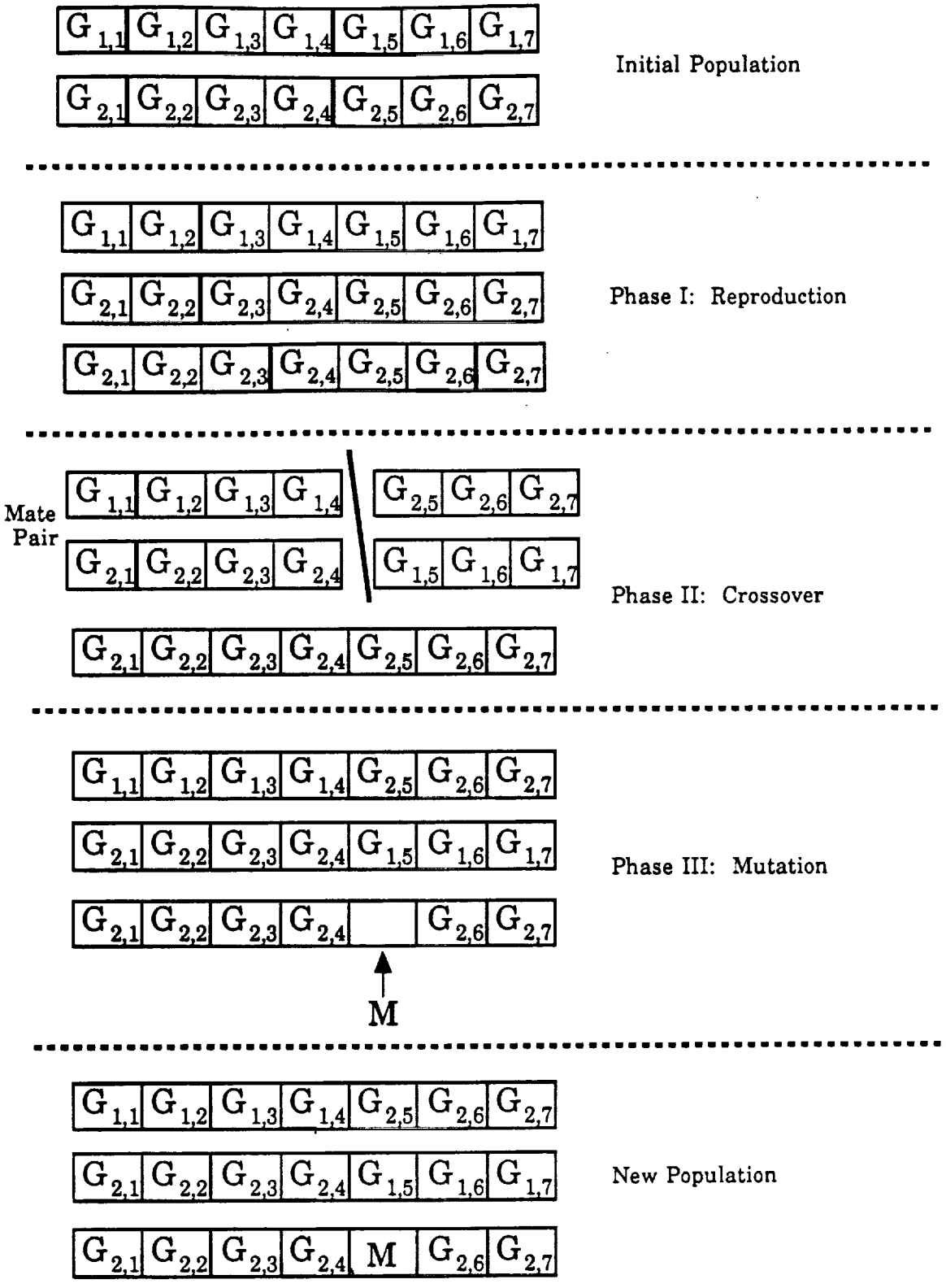


Figure 4-1. Sample Generation of Genetic Algorithm Model

Genetic Algorithm Specifications

An example using Goldberg's algorithms (1989) is shown in figure 4-2 to illustrate the details of how a GA model operates. Strings in the current generation are maintained in a population database. De Jong (1990) illustrated that binary-valued genes promote effective search processes in a GA model. Thus, the strings are represented as 1's and 0's. Furthermore, Grefenstette (1990) suggested a population size of between 50 to 100 strings to insure satisfactory performance. Therefore, for the sake of consistency, all genetic models developed for the RTT had a population size of 50. For the sake of simplicity, the example has a population of only four strings.

The GA model first evaluates the strings to fitness values. Figure 4-3 illustrates how the fitness value is calculated. A string, say X, is first decomposed to components. The components are then collectively evaluated by an objective function, f. The output of the function is the fitness value of string X. A probability count, based on the fitness value, is then calculated for each string.

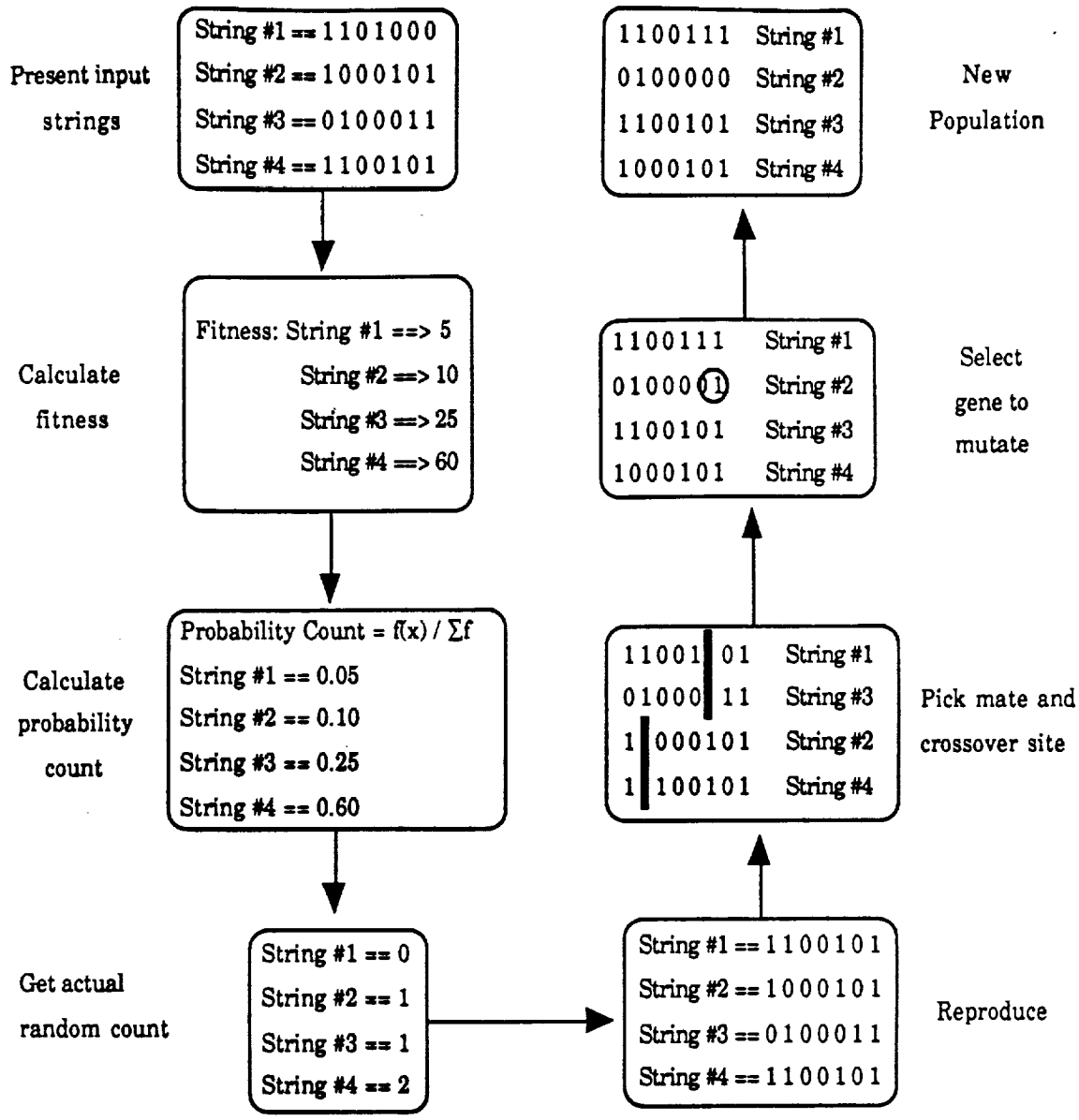


Figure 4-2. Sample Genetic Algorithm

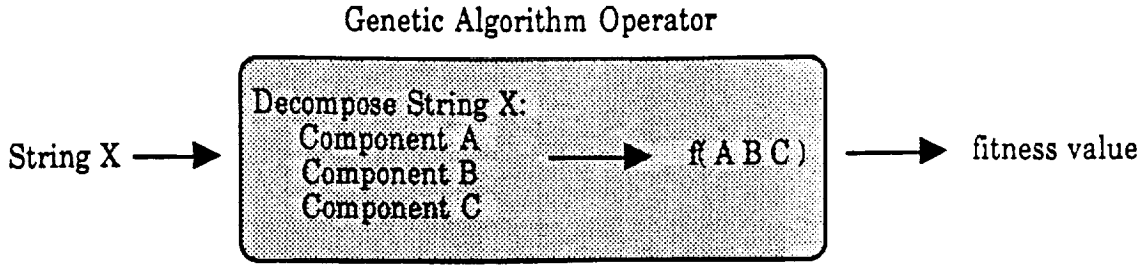


Figure 4-3. Fitness Value Evaluation Process

The probability count is then mapped onto a "roulette wheel" (figure 4-4).

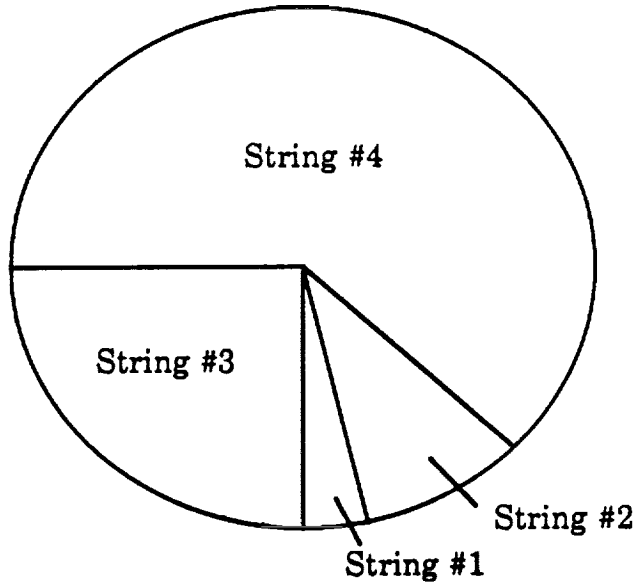


Figure 4-4. Roulette Wheel Based on Example

For N strings in the population, the roulette wheel is spun N times. The number of times a particular string is chosen by the wheel, denoted as the random count, indicates the number of copies of itself that will exist in the next generation. The entire process from calculating fitness values to generating random counts is known as reproduction.

Following reproduction, the crossover operator is applied to the strings. Two conditions must be satisfied in order to perform crossover.

First, pairs of mates must be chosen randomly from the population. Second, a crossover site must be randomly selected for the mating pair. Once the sites are selected (shown in figure 4-2 as between genes 5 and 6 for the first pair and between genes 1 and 2 for the second pair), all the gene values of the string pairs after the crossover site are switched.

Following crossover, the strings are then manipulated by the mutation operator. The mutation operator randomly (with small probability) alters the value of a gene. In the example, string 2 has been chosen to be mutated at genes 6 and 7. De Jong (1990) suggested that the mutation probability should be less than 0.001. The resultant population, after undergoing reproduction, crossover, and mutation, represents the next generation of strings.

GA Models of Cognition

Goldberg (1989) offered a review of GA models in the social sciences. However, the models reviewed were normative in nature. A normative model prescribes what an individual should do as opposed to what he/she actually does. The papers reviewed included a model of prehistoric hunter-gatherer behavior (Reynolds, 1979) and a model to solve an iterated prisoner's dilemma problem (Axelrod, 1985). Grefenstette (1990) also provided a brief review of GA applications. The topics he described were: numerical function optimization; optimization of computer simulated processes; combinatorial optimization; image processing; game playing; and multiobjective pattern classification. In light of the literature reviewed, the trend of most GA-related research appears to be oriented

towards optimization. Nevertheless, functional aspects of a GA model--such as avoidance of local minima, and new rule generation--can be applied to simulate satisficing behavior as well.

GA Model of Perceptual Decision-making

To overcome the search restrictions (trap of local minima) and representation limitations (inability to create new features) of the ANN model of the RTT, a GA model was constructed. Although slight variations of the genetic operators were needed to create a model for the RTT, the general function of the model remained the same as the one illustrated in figure 4-2. The first modification involved the representation of the strings, which were interpreted as rules. Each string represented the operation:

H^a **operator** W^b

where **H** was the height of the rectangle, **W** was the width of the rectangle, **a** and **b** were the exponents on the height and width, and **operator** was one of the arithmetic operators (+,-,x,÷). In terms of the objective function evaluation process shown in figure 4-3, **a**, **b**, and **operator** were all string components. The interpretations of the rules, i.e. f(**a b operator**), dictated an ordering of the four displayed rectangles in the RTT. The string itself was represented as:

aa bb operator

Thus, for example,

10 01 111

represented the operation

$H^2 \div W^1$

where $H^2 + W^1$ is the criterion determined by the string **10 01 111** to order input rectangles. Note that **a**, **b**, and **operator** were all represented in binary. Although some representations of the strings were not perceptually available, such as $H^2 + W$ or $1 + W^2$, others, such as shape ($H + W$) or area ($H \times W$) were.

A second modification involved the methods of crossover and mutation. Figure 4-2 showed crossover occurring for all genes following the crossover site. Sanderson et al. (1989) showed that humans are attuned to invariants within a perceptual environment. Therefore, to effectively simulate human performance, the GA model should also seek and maintain invariant cues. In Goldberg's algorithm, all genetic information after the crossover site were exchanged between mates, and all information after the mutation site was altered. To do so with the GA model of the RTT, however, would risk breaking the integrity of **a**, **b**, and **operator** components. For example, if the crossover site between two mates was selected to be between genes 1 and 2, only half of the **a** component information would be exchanged. Thus, possible invariant cues, such as **a**, **b**, and **operator** components, would be lost due to random site selection. To maintain component integrity, crossover and mutation sites were preselected. These sites were between **aa** and **bb** genes, and between **bb** and **operator** genes.

Introduction to Genetic-based Machine Learning Methodology

A major drawback of the GA model is its instability. Given that the number of strings in a population remains constant, each generation

contains a different set of strings. For the RTT, each rectangle pattern represented a new generation. Thus, if a string scored poorly on one displayed rectangle set, but was the best performer overall, it could have been eliminated from the population because of the poor score. To prevent such occurrences, an alternative model was considered. That model, called the genetic-based machine learning (GBML) model (Goldberg, 1989), is a rule-base system using genetic algorithm search techniques.

A GBML follows the form:

if <condition> then <action>

Incoming messages from the environment are detected by "classifier" strings. These strings, using the same representation as GA strings, classify inputs by competition with one another. Classification is accomplished, i.e. <condition> is met, when a string has won possession of the input message. The <action> part of the rule-base is determined by the winning string. As in the GA example, the GBML model also contains a string population. Each string contains a strength value. The strength of a string, unlike the fitness value, is an internal evaluation of its worth. Unlike the GA model, where a poor fitness value could eliminate a string from the population, the GBML model has a "second line of defense." The internal evaluation keeps a running count of the performance of a string through a number of generations. After a predetermined number of generations, the weaker strings are replaced with stronger ones. Therefore, a gradual replacement based on strength of the string population occurs.

A GBML model consists of two major components: the apportionment of credit (AOC) system, and GA. The AOC system uses a bucket brigade algorithm (BBA) (Goldberg, 1989). The BBA contains two main components: an auction and a clearinghouse. The BBA can be viewed as an economic system where information is traded from the manufacturer (inputs from the environment) to consumers (the strings). In the auction, strings which qualify (those that match bidding preconditions) bid a portion of their strength for a message. The winning string, the one with the highest bid, then pays its bid to the previous owner of the message, and becomes the new owner. The bidding continues until no remaining strings are qualified to bid. At this point, the string which holds the message is evaluated by the clearinghouse. Based on the quality of the message, the string is either rewarded or punished by increasing or decreasing its strength. Each auction-clearinghouse cycle represents a generation.

The GBML model executes the GA component after a predetermined number of generations of the BBA component. Thus, unless the GA component is executed after every generation of the BBA, the GBML model should be more stable than the GA model.

GBML Models of Cognition

A survey of existing GBML models shows that although some model characteristics show similarity to human cognitive processes, the models are normative in nature. Spiessens (1990) described a GBML model which was able to predict objective function information based on an internal

world model. The model was used to solve a state space search problem. The description of the model is similar to some psychological descriptions of mental models (Richards, 1990). Riolo (1989) described the emergence of default hierarchies in some GBML models. The models were classifier systems used to solve a categorization task. The hierarchies seems similar to hierarchical phases of skill, ranging from controlled behavior to automatic behavior (Schneider and Detweiler, 1988). In spite of the similarities, the trend of GBML research, as in GA research, tends toward optimization. Nevertheless, characteristics of the GBML model behavior--such as population stability and sub-optimal, but satisficing, performance--can be applied to model human decision-making performance.

GBML Model of Perceptual Decision-making

Although slight modifications to the BBA were needed in order to construct a GBML model of perceptual decision-making, the basic concept of the BBA remains the same. In the BBA, each input message from the environment is important to a particular set of strings (those which are qualified to bid). The input representation for the rectangles of the RTT, however, does not reveal preferences for rule strings in the population. Therefore, to compensate for the discrepancy, four adaptations were made in the GBML model:

1. The input message was regarded as a dummy token representing dimensions of each displayed rectangle set. Unlike the BBA, the message in the GBML model for the RTT had no meaning to specific strings.

2. Each message was traded only once, and all strings were qualified to bid on all messages.
3. The winning string of each auction paid its bid to the clearinghouse. Once the string was evaluated to a RTT payoff score the clearinghouse rewarded or punished the string based on the a payoff score threshold.
4. The GA component was executed after all displayed rectangle patterns were presented to the BBA component.

An overview of the model is provided in figure 4-5.

A hypothetical example iteration of the AOC portion of the GBML model for the RTT is shown in figure 4-6. The inputs into the system, regarded as tokens, are auctioned to the highest bidding rule (rule 3). Once the rule wins, it pays the bid from its strength. The rule is then allowed to be evaluated to a payoff score of, say 70.0. Since the score is lower than the threshold, the rule is punished for poor performance by subtracting the amount of bid from its strength. At the end of the iteration, the strength of rule 3 decreases to 64.0.

The modified BBA (MBBA) is repeated for each RTT rectangle pattern. Following the MBBA, genetic operations are applied to the strings. The strength value of the strings now becomes the fitness value for the GA. At this point, the fitness value represents an aggregate measure of the worth of each string over all displayed rectangle patterns. The GA model for the GBML is the same as the one used solely for the RTT.

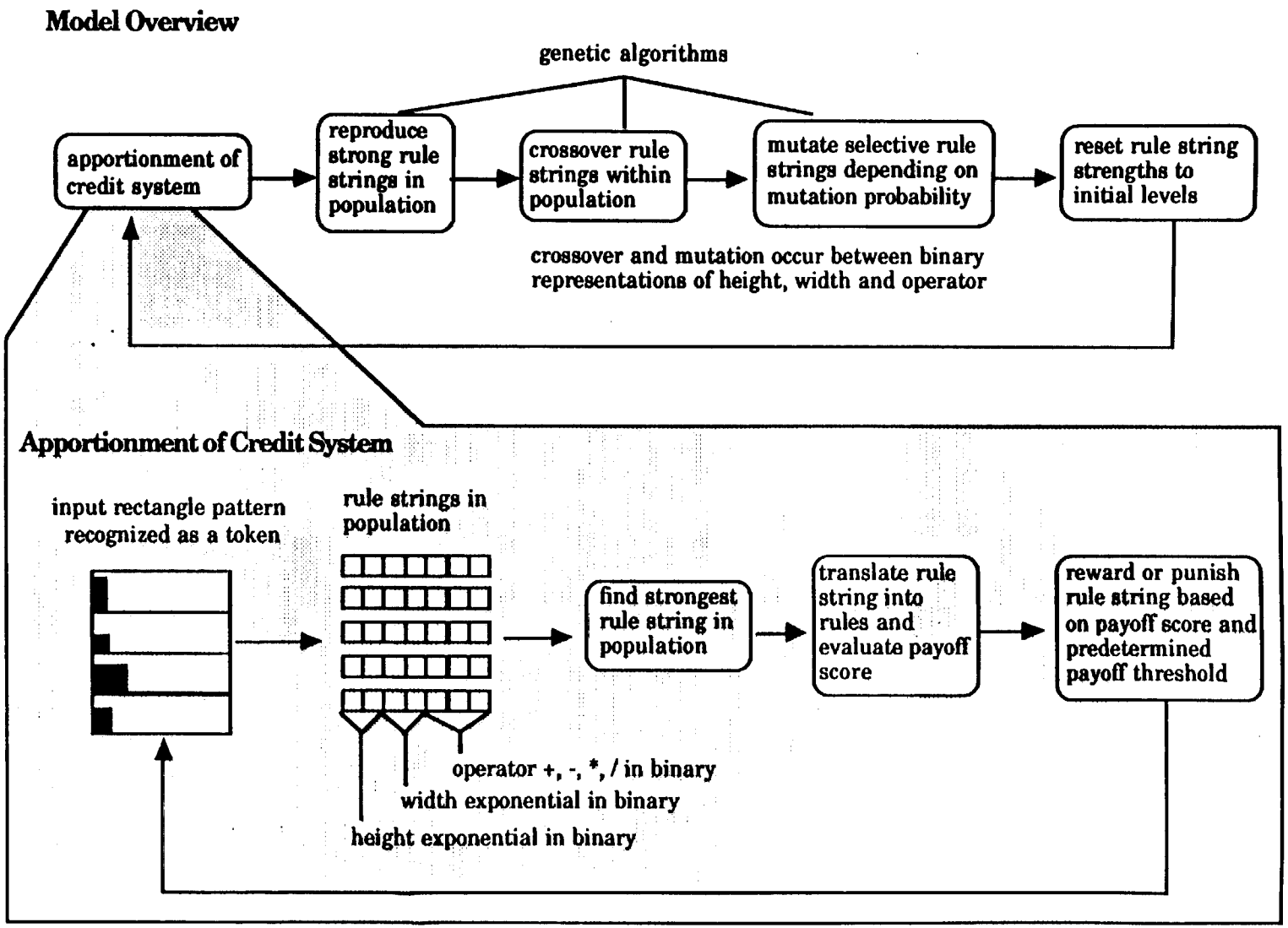


Figure 4-5. Genetic-Based Machine Learning Model

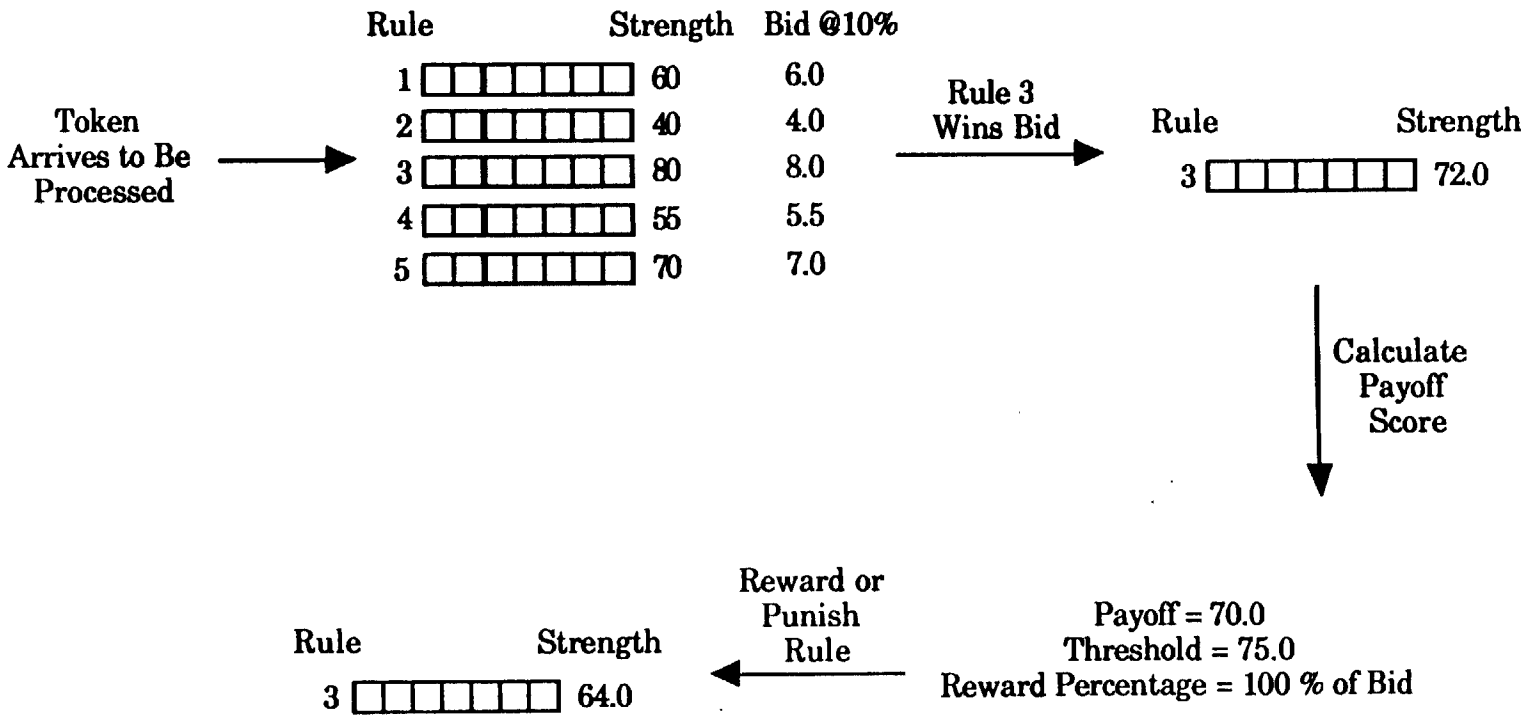


Figure 4-6. Sample Apportionment of Credit Iteration

Comparison of GA and GBML Methodologies

Schuurmans, Chai, and Shu (1987) observed that a classifier system, such as GBML, appears to "lack the killer instinct." That, although the system often reaches near-optimal solutions, global maxima are infrequently reached. In the case of human decision-making, however, flexibility and "non-optimal" methods might actually make better models. De Jong stated:

> The current popular view is that the classifier approach will prove to be most useful in on-line, real-time environments in which radical changes in behavior cannot be tolerated; whereas the [GA] approach will be useful with off-line environments in which more leisurely exploration and more radical behavioral changes are acceptable.
> (De Jong, 1990, p. 628)

By modeling perceptual decision-making using both the pure GA and the GBML methodologies, perhaps some inferences concerning human decision-making could be made. Maybe the radical behavior of the pure GA model is indicative of human performance. Or, perhaps the conservative approach of the GBML model is more akin to human performance.

CHAPTER V

COMPUTER IMPLEMENTATION

Computer Program Design

Before the mathematical models could be coded into computer programs, an overview of the flow of information in the perceptual decision-making paradigm was needed. Once an overview was provided, the specifications of the backpropagation algorithm (discussed in Chapter III), the GA (discussed in Chapter IV), and the GBML (also discussed in Chapter IV) models could be programmed so that a correspondence between the models and the decision-making paradigm became apparent. Figure 5-1 shows an information flow diagram for the perceptual paradigm. The diagram contains two types of processes: purely perceptual (initial stages of processing linked by a solid line) and training (stages of processing following the dotted line). Purely perceptual processing occurs when perception of cues in an environment directly evokes an action. When purely perceptual processing is not possible, training will be required to search for more diagnostic cues. Training involves determining the significance of feedback, and based on that significance, adjusting the amount of attention given to different environmental cues.

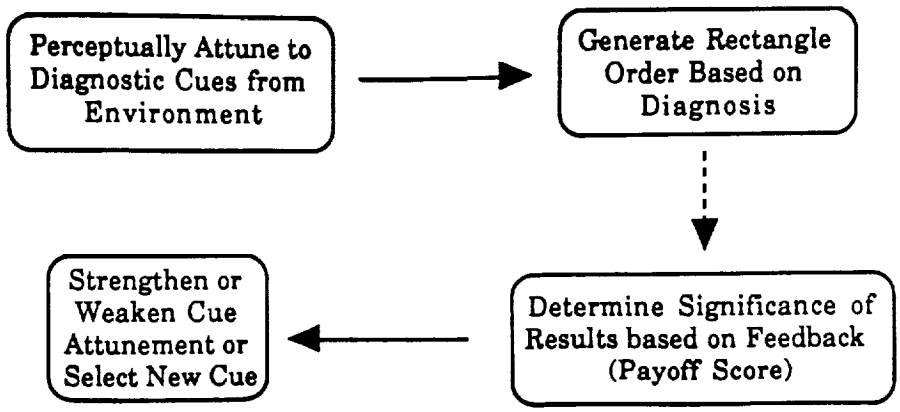


Figure 5-1. Information Flow in the Perceptual Decision-making Paradigm

A computer implementation of the perceptual paradigm using an ANN model is shown as an information flow diagram in figure 5-2. The forward data path of the ANN model represents purely perceptual processing. Input representations and fixed subject reaction times are first processed by the input layer and passed forward to hidden and output layers via the weights. The strengths of the output layer nodes are then converted to an ordering of displayed rectangles. A payoff score is then calculated using the ordering. The feedback data path of the ANN model represents training. The ordering of displayed rectangles is compared to a target order. Errors from the comparison are propagated back through the ANN model and are used to update existing weights.

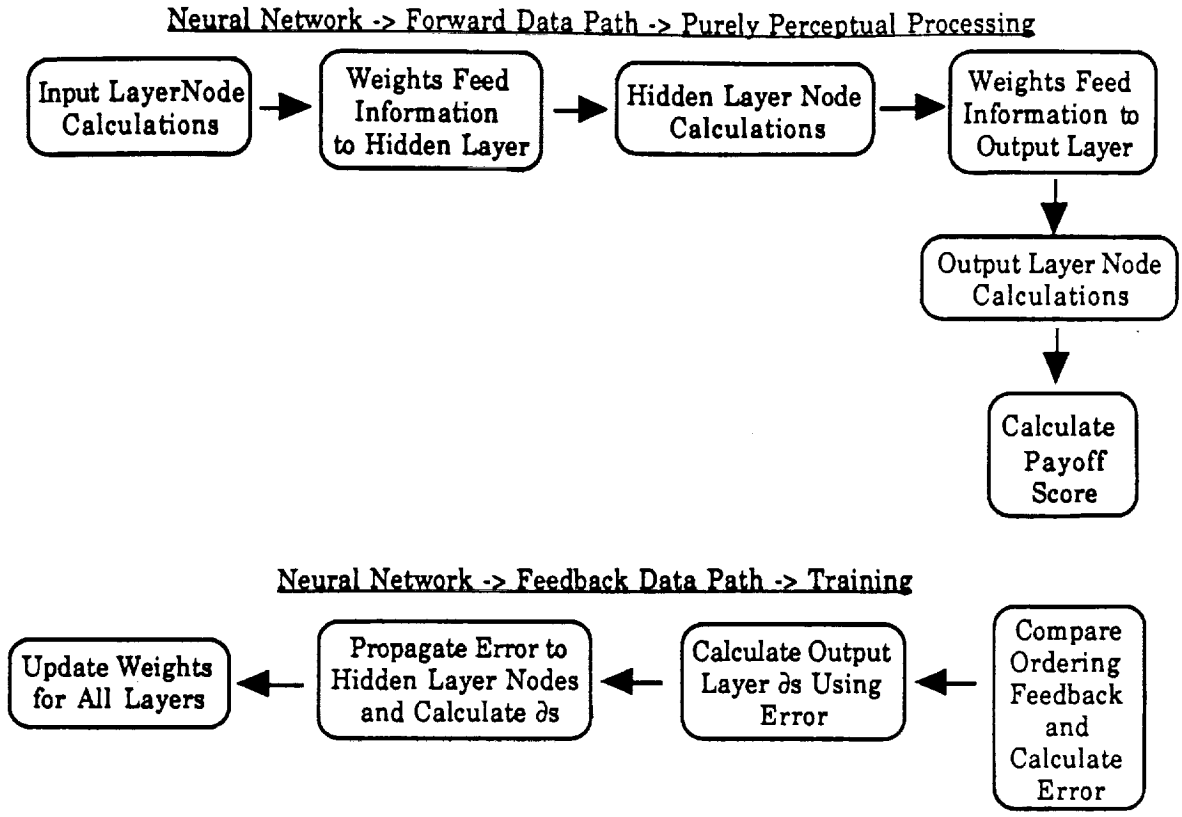


Figure 5-2. ANN Model of the
Perceptual Decision-making Paradigm

A program of the perceptual paradigm using a GA model is depicted as an information flow diagram in figure 5-3. As in the ANN program, the forward data path represents purely perceptual processing. Input representations and fixed subject reaction times are first detected and stored into a GA database. Each string in the database is then translated in a rule, and the rule is activated to generate an ordering of displayed rectangles. A payoff score is then calculated using the ordering. Unlike the ANN program, which calculates one payoff score per displayed rectangle pattern, the GA program calculates 50 payoff scores (one for each of the 50 rules in the population). The feedback data path of the GA

represents training in the perceptual paradigm. Rule strings registering higher payoff scores are reproduced to replace weaker strings. Strings are then coupled into 25 pairs by the crossover component. After the exchange of information occurs within string pairs, strings are selected, at random, to be mutated.

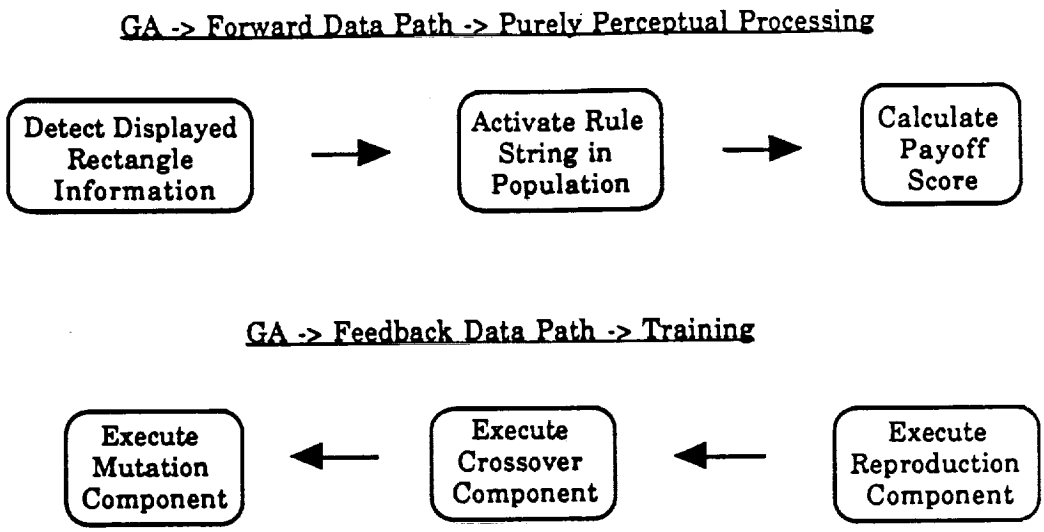


Figure 5-3. GA Model of the
Perceptual Decision-making Paradigm

A computer implementation of the perceptual paradigm using a GBML model is shown as an information flow diagram in figure 5-4. The GBML model uses the same forward data path as the GA model. Unlike training for the perceptual paradigm, where each displayed rectangle pattern is sequentially trained, training for the GBML model resemble a "batch" mode of processing. The processes shown within the shaded region in figure 5-4 represent the apportionment of credit (AOC) system. All displayed rectangle patterns are first processed by the AOC. As rule

strings bid for the displayed rectangles, an internal value system tracks the worth of the strings based on the goodness of payoff scores. However, no training (modifications to the rule strings) occurs in the AOC system. The reproduction, crossover, and mutation components, as described in the GA model, actually modify the rule strings. Thus, since the GA components are not executed until after AOC processes, the rule strings are essentially modified in terms of feedback from processing a batch of displayed rectangle patterns.

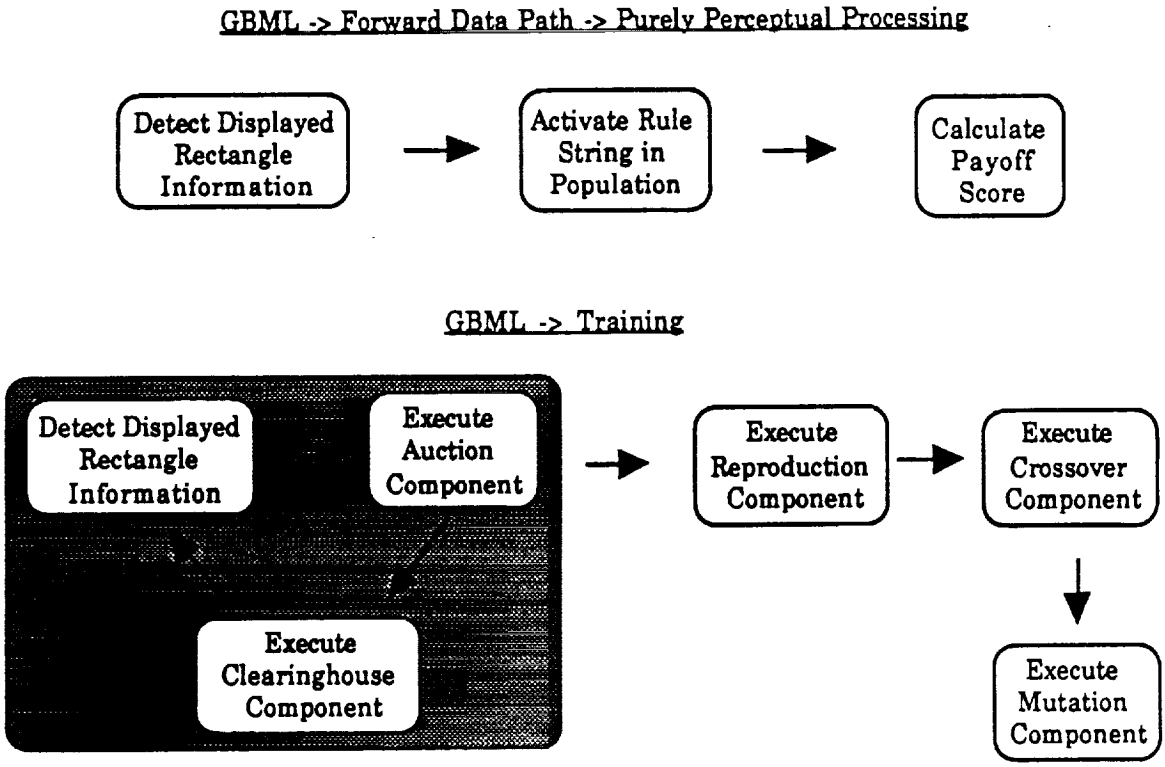


Figure 5-4. GBML Model of the Perceptual Decision-making Paradigm

Artificial Neural Network Program Description

Using the ANN model flow diagram in figure 5-2, an object-oriented design was constructed. A pseudo-code description of the ANN program is given in Appendix 1. Figure 5-5 illustrates the encapsulation structure.

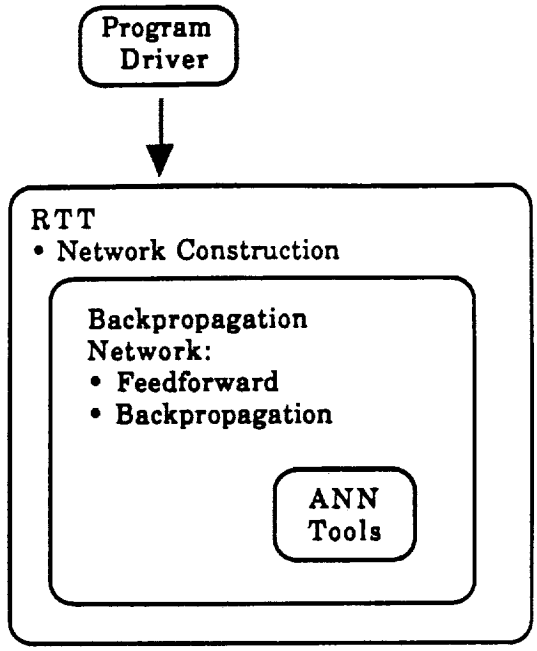


Figure 5-5. Artificial Neural Network Program Structure

The program driver controls the flow of the ANN program. The RTT module provides the method to construct feedforward and backpropagation networks. The module also includes an initialization procedure for the program. The Backpropagation Network module, a subclass of the RTT module, contains components of a backpropagation network. Also included in the module are components of a network using the skeletonization procedure. Components in the Backpropagation Network module represent operations on a particular layer of nodes. Encapsulated within the Backpropagation Network module, the ANN Tools module provides

elemental components of a network. Components in the ANN Tools module provides the method to construct a node and a weight.

Genetic Program Description

Since the GA model can be considered as a part of the GBML model, the two models were implemented in one program. Using the GA and GBML model flow diagrams in figures 5-3 and 5-4, an object-oriented program was constructed. The pseudo-code of the program is shown in Appendix 2. Figure 5-6 shows the programming structure.

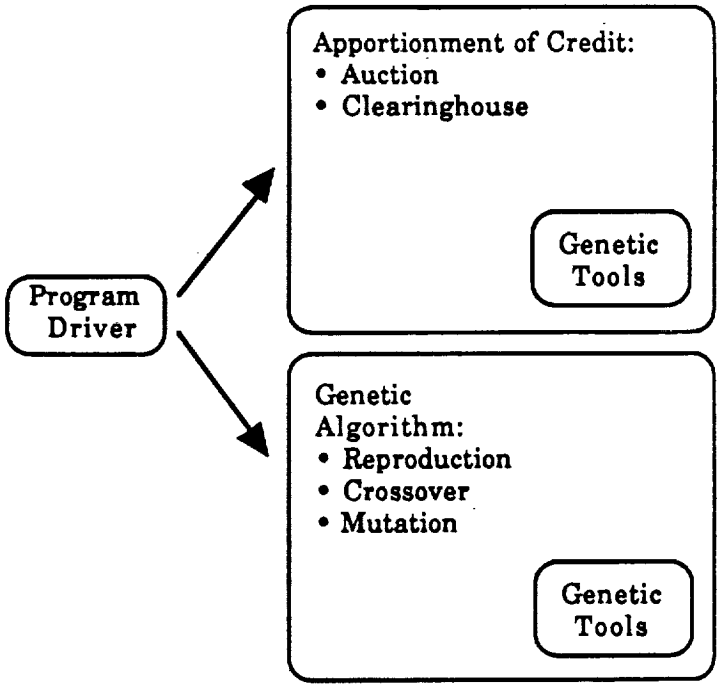


Figure 5-6. Genetic Program Structure

The program driver controls the flow of the program. If the GA model is selected, only reproduction, crossover, and mutation components are executed. However, if the GBML model is selected, the AOC would first be executed for all input displayed rectangle patterns. Then, the GA module

would be processed. Both the GA and GBML modules encapsulate the Genetic Tools module. The tools module provides the means to construct rule strings and input displayed rectangle representations.

Computer Program Options

Tables 5-1 and 5-2 provide a nomenclature for the different variations of the programs. The variations were needed to show the effectiveness of the ANN, GA, and GBML models to perform optimally, and to simulate subject performance. There were three areas of variability: input representation, subject time data, and mode of operation.

For the ANN program, the dimensional input representation contained 20 real-valued input nodes reflecting the height, width, area, shape, and height^2 + width dimensions of each displayed rectangle set (see figure 4-4). The relational representation contained 20 binary-valued input nodes representing the rectangles in each displayed set with the largest dimensional values. The pair-wise relational representation contained 30 binary-valued input nodes showing the pair-wise comparisons of dimensional values for all rectangles in each displayed set. To test the effectiveness of the ANN model, different modes of operation were necessary. Training on the optimal order (in which the target node configuration represented the optimal order) could provide a cursory look at the ability of the model to learn. If the model could not be trained in this mode, further attempts to train on subject orders will be fruitless. Training on the subject order (in which the target node configuration represented the subject's order) provided a measure of model effectiveness to simulate

		Input Representation				Time Data					Mode of Operation		
	Model	Dimensional	Relational	Pair-wise Relational		Subject 1 Reaction Time	Subject 2 Reaction Time	Subject 3 Reaction Time	Subject 4 Reaction Time		Train on Optimal Order	Train on Subject Order	Predict
ANN Models	ANN_A	X				X					X		
	ANN_B		X			X					X		
	ANN_C			X		X					X		
	ANN_D	X				X							X
	ANN_E		X			X							X
	ANN_F			X		X							X
	ANN_G	X					X						X
	ANN_H		X				X						X
	ANN_I			X			X						X
	ANN_J	X						X					X
	ANN_K		X					X					X
	ANN_L			X				X					X
	ANN_M	X							X				X
	ANN_N		X						X				X
	ANN_O			X					X				X
	ANN_P	X				X						X	
	ANN_Q		X			X						X	
	ANN_R			X		X						X	
	ANN_S	X					X					X	
	ANN_T		X				X					X	
	ANN_U			X			X					X	
	ANN_V	X						X				X	
	ANN_W		X					X				X	
	ANN_X			X				X				X	
	ANN_Y	X							X			X	
	ANN_Z		X						X			X	
	ANN_AA			X					X			X	

Table 5-1. Nomenclature for ANN Models

		Input Representation				Time Data					Mode of Operation		
	Model	*Dimensional*	*Relational*	*Pair-wise Relational*		*Subject 1 Reaction Time*	*Subject 2 Reaction Time*	*Subject 3 Reaction Time*	*Subject 4 Reaction Time*		*Train on Optimal Order*	*Train on Subject Order*	*Predict*
GA Models	GA_A	X				X					X		
	GA_B	X					X				X		
	GA_C	X						X			X		
	GA_D	X							X		X		
	GA_E	X				X							X
	GA_F	X					X						X
	GA_G	X						X					X
	GA_H	X							X				X
	GA_I	X				X						X	
	GA_J	X					X					X	
	GA_K	X						X				X	
	GA_L	X							X			X	
GBML Models	GBML_A	X				X					X		
	GBML_B	X					X				X		
	GBML_C	X						X			X		
	GBML_D	X							X		X		

Table 5-2. Nomenclature for GA and GBML Models

su

bject performance. Once trained, an ANN model can be switched to prediction mode to predict outcomes for novel input representations.

For the GA and GBML models, the input representation contained the height and width dimensions of each displayed rectangle pattern. For both genetic models, training on optimal order (in which the RTT payoff score served as the learning criteria) provided a glance at the capabilities of the models. Once the models showed that learning was possible, training was switched to subject order. Since the learning criteria for both genetic models was a scalar objective function value, a translation function was used to convert the subject order to a fitness measure. The prediction mode of operation for both genetic models provided rule string translations to predict payoff outcomes for input representations.

For the ANN, GA, and GBML models, variations in reaction times were assigned across different input representations and modes of operation to account for variability between the four subjects.

CHAPTER VI

DATA ANALYSIS

Selection of Criterion

To determine the adequacy of the ANN, GA, and GBML models to simulate human performance, a standard of measurement was needed. Since the RTT payoff score can readily be interpreted, it was first tested as the metric. To sample the performance of the ANN models, three models (shown in table 5-1 as ANN_A, ANN_B, and ANN_C) were trained for 1 epoch on 320 displayed rectangle patterns. To maintain consistency, one subject's reaction time (Subject 1) was used for all three models. Figure 6-1 shows Subject 1's payoff score profile for 320 patterns (representing sessions on days 2-5). Figure 6-2, 6-3, and 6-4 show payoff profiles generated by ANN models.

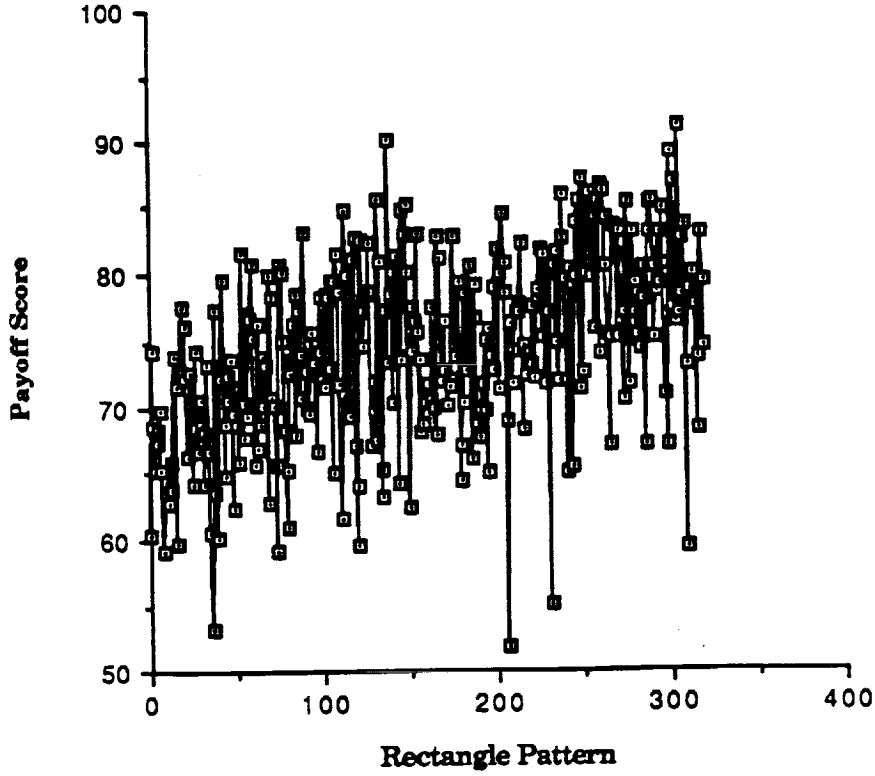


Figure 6-1. Subject 1 Payoff Profile

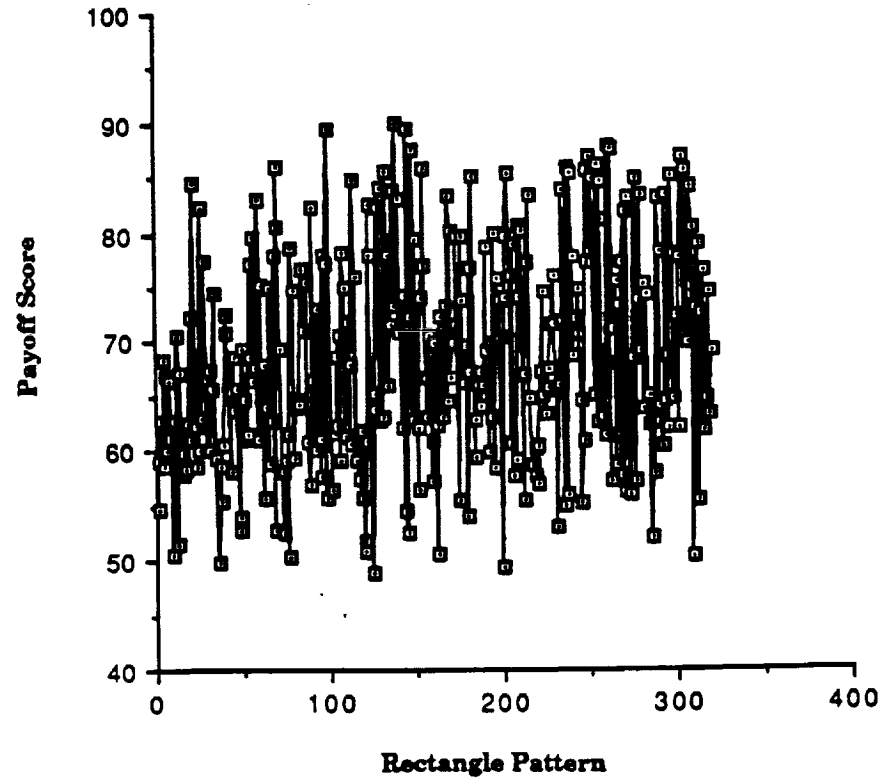


Figure 6-2. Model ANN_A Payoff Profile Using Subject 1 Reaction Times

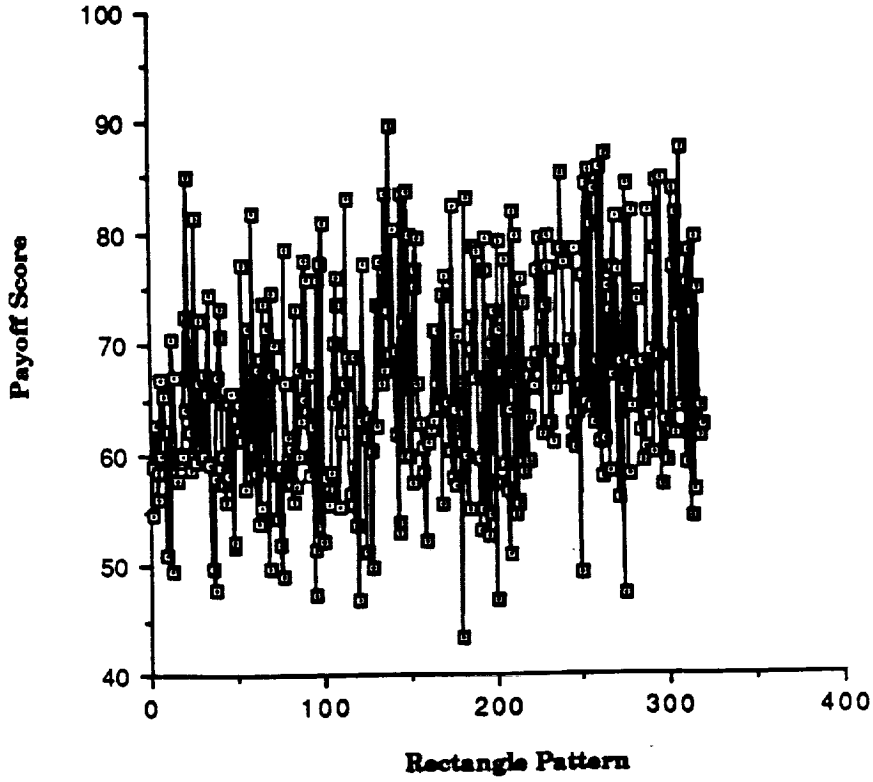


Figure 6-3. Model ANN_B Payoff Profile Using Subject 1 Reaction Times

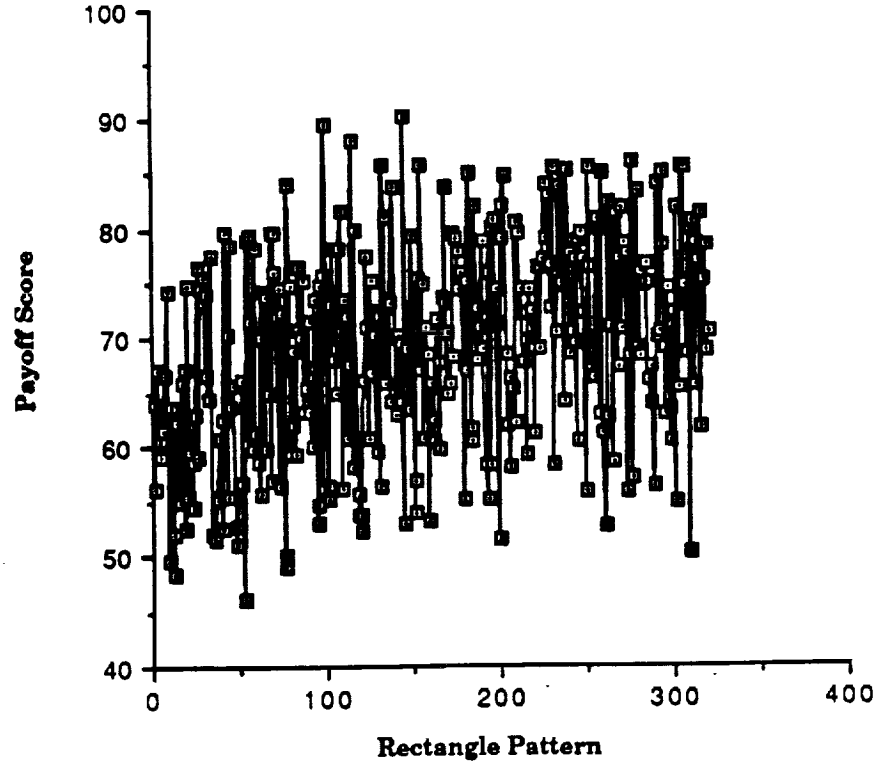


Figure 6-4. Model ANN_C Payoff Profile Using Subject 1 Reaction Times

Observation of figures 6-1 to 6-4 revealed a learning curve for all payoff score profiles. Initial scores, which ranged between 50 and 70, improved to between 70 and 90. The learning curve appeared to indicate an improvement in the ability to order the rectangles as the subject or model gained experience. However, recall that equation *(2.2)* showed that reaction time (delay time) contributed to the payoff score calculation. Subject 1's reaction time profile, shown in figure 6-5, indicates a decrease in reaction time as the practice patterns increased. Thus, since both the ordering of the rectangles and the reaction time affect the payoff score, an accurate assessment of model performance could not depend on the payoff score alone.

To illustrate the problem, consider the effects of reaction times given that the rectangles are optimally ordered for each trial of a session versus the same reaction times given that the rectangles are worst ordered (opposite optimal order) for the same session. Using Subject 1 reaction times, the optimal ordered case generates an average payoff score of 78.42 for 320 trials while the worst ordered case produces an average payoff score of 56.95. Recall that the payoff scores can range between 0 and 100. For models using Subject 1 reaction times, however, the payoff mean is further restricted between 56.95 and 78.42. Because each subject's reaction time profile is different, the range for payoff means is also different. Thus, the meaning of error (the difference between a target score and a model generated score) is relative to each subject's reaction time profile.

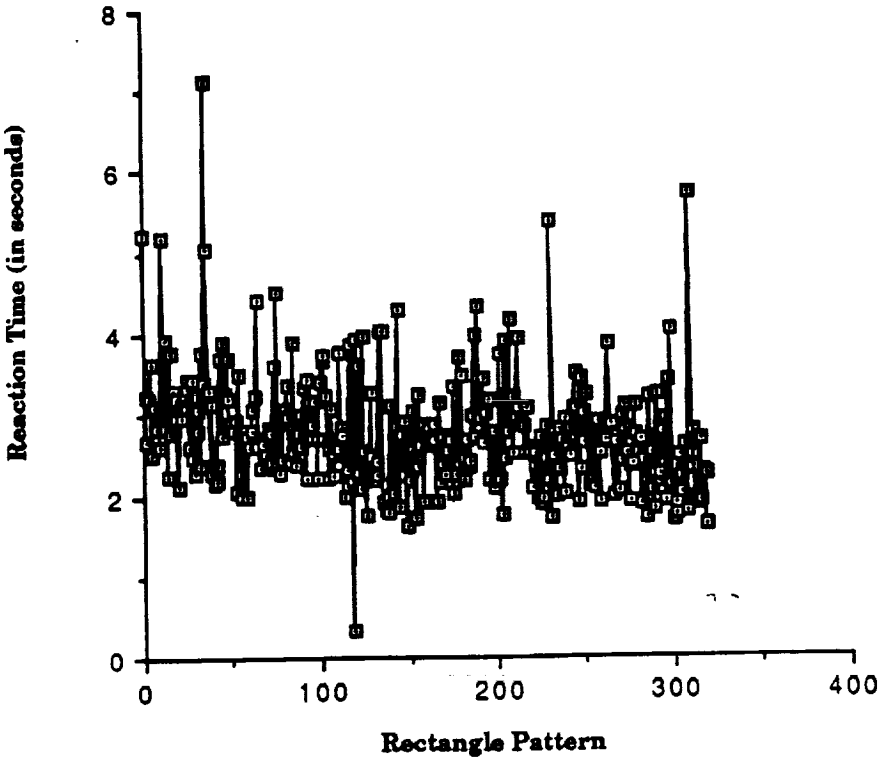


Figure 6-5. Subject 1 Reaction Time Profile

To successfully determine model effectiveness across subjects, therefore, a different metric was required. The metric needed to be independent of subjects. Such a metric was found in the offset, or error, of a generated order from a target order. For instance, the generated order:

4 3 1 2

and the target order:

3 4 1 2

creates an offset of two.

The offset metric represents one way of measuring model performance. Some loss of information, however, is possible through its use. For instance, the difference of the orders:

4 3 1 2 and **3 4 1 2**

represent a slight spatial error--neighboring rectangles are misordered. Whereas the difference of the orders:

4 1 2 3 and **1 4 2 3**

show a much larger spatial error. Nevertheless, the offset metric was chosen because it provided the simplest measure of model performance.

ANN Model Analysis

Prior to modeling subject performance in the RTT, the ANN models must first show the ability to learn the task. Using Subject 1 reaction times as a testbed, models ANN_A, ANN_B, and ANN_C were trained on 240 rectangle patterns (days 2-4) for 20000 epochs using random initial weights. Note that although the amount of training was greater for the model as opposed to the subject, the objective of this research was not to generate

process models. The trained weights were then used by models ANN_D, ANN_E, and ANN_F to predict outcomes for displayed rectangle patterns on the final session (day 5). To test the effects of trained versus untrained weights, models ANN_D, ANN_E, and ANN_F also used the random initial weights to predict outcomes for day 5 displayed rectangle patterns. The mean scores for outcomes of models ANN_D to ANN_F are shown in table 6-1. Since the analysis was done across input representations instead of subject reaction times, the payoff score was used as the standard of comparison.

Subject	Representation: Dimensional	Representation: Relational	Representation: Pair-wise Relational
1	Model: ANN_D Trained Weights: Mean Payoff = 76.31 Untrained Weights: Mean Payoff = 69.24	Model: ANN_E Trained Weights: Mean Payoff = 76.11 Untrained Weights: Mean Payoff = 69.25	Model: ANN_F Trained Weights: Mean Payoff = 76.26 Untrained Weights: Mean Payoff = 67.80

Table 6-1. Artificial Neural Network Payoff Results after 20000 Training Epochs on Optimal Order

An analysis of variance (ANOVA) table was constructed to show the effects of training in models ANN_D, ANN_E, and ANN_F. Table 6-2 shows a two-way ANOVA with Training (trained or untrained weights) as one effect and Representation (dimensional, relational, and pair-wise

relational) as the other effect. The ANOVA model is as follows:

$$Payoff_{ijk} = \mu + Training_i + Representation_j + (Training*Representation)_{ij} + error_{ijk}$$

for i=1,2; j=1,2,3; and k=1,...,80.

where μ is the population mean of the Payoff and (Training*Representation) is the interaction effect of training and input representation. The analysis found the training effect to be highly significant, Pr{F(5,474)>84.49}=0.0001. Thus, training was effective in improving payoff scores across all three ANN models.

ANOVA Procedure

Dependent Variable: Payoff

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	6764.2609939	1352.8521988	17.17	0.0001
Error	474	37347.5677842	78.7923371		
Corrected Total	479	44111.8287780			

Source	DF	SS	Mean Square	F Value	Pr > F
Training	**1**	**6657.3972301**	**6657.3972301**	**84.49**	**0.0001**
Representation	2	49.6538772	24.8269386	0.32	0.7299
Training*Representation	2	57.2098865	28.6049432	0.36	0.6958

Table 6-2. Two-way ANOVA of Training for Optimal Order.

Analysis of ANN Models of Subject Performance

Once the ANN models were shown to be capable of learning, the models were then applied to simulate subject performance. To increase fidelity to subject training conditions, 12 models (ANN_P to ANN_AA) were trained for one epoch using displayed rectangle patterns for sessions on days 2-4. The 12 models used for the 4 different subject reaction time profiles and the 3 different input representations. The trained weights

from the models were then used by models ANN_D to ANN_O to predict performances for day 5 displayed rectangle patterns. To test the effect of trained versus untrained weights, initial random weights were also used by models ANN_D to ANN_O to predict outcomes for day 5 patterns. The difference between offsets generated by models using untrained weights and trained weights were then compared across subject reaction times and input representations. Thus, the amount of offset difference measured a model's learning capability. The mean of offset difference is calculated as follows:

$$\text{Offset\_difference\_mean} = \Sigma\ (\text{untrained\_model}_i - \text{trained\_model}_i) / 80$$

for i=1,...,80

where untrained_model_i is the number of offsets generated by the model using untrained weights for displayed rectangle pattern i, and trained_model_i is the number of offsets generated by the model using trained weights for pattern i. Thus, a positive mean of difference value signifies that learning has occurred. A negative mean of difference value, however, indicates that the model has not learned. Table 6-3 shows the mean of the offset differences per displayed rectangle pattern for each of the models.

Subject	Representation		
	Dimensional	Relational	Pair-wise Relational
1	Model: ANN_D Mean of Difference*: 0.075	Model: ANN_E Mean of Difference: -0.025	Model: ANN_F Mean of Difference: 0.575
2	Model: ANN_G Mean of Difference: 0.125	Model: ANN_H Mean of Difference: 0.050	Model: ANN_I Mean of Difference: 0.288
3	Model: ANN_J Mean of Difference: 0.150	Model: ANN_K Mean of Difference: -0.138	Model: ANN_L Mean of Difference: 0.350
4	Model: ANN_M Mean of Difference: 0.088	Model: ANN_N Mean of Difference: 0.075	Model: ANN_O Mean of Difference: 0.400

* Mean of difference between results of model using untrained weights and model using trained weights.

Table 6-3. Artificial Neural Network Offset
Results after 1 Training Epoch on Subject Order

An ANOVA table was constructed to determine the significance of the offset differences in models ANN_D to ANN_O. Table 6-4 shows a two-way ANOVA with representation and subject reaction time (Subject 1, Subject 2, Subject 3, and Subject 4) as the two effects. The ANOVA model is as follows:

$$\text{Offset\_difference}_{jmk} = \mu + \text{Representation}_j + \text{Subject}_m + (\text{Representation}*\text{Subject})_{jm} + \text{error}_{jmk} \qquad (6.1)$$

for j=1,2,3; m=1,2,3,4; and k=1,...,80.

The analysis found the Representation effect to be significant, $Pr\{F(2,948)>6.75\}=0.0012$.

ANOVA Procedure

Dependent Variable: Offset_difference

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	**11**	**34.98645833**	**3.18058712**	**1.49**	**0.1304**
Error	948	2027.01250000	2.13819884		
Corrected Total	959	2061.99895833			

Source	DF	SS	Mean Square	F Value	Pr > F
Representation	**2**	**28.85833333**	**14.42916667**	**6.75**	**0.0012**
Subject	3	1.06145833	0.35381944	0.17	0.9196
Representation*Subject	6	5.06666667	0.84444444	0.39	0.8825

Table 6-4. Two-way ANOVA of Training on Subject Order for 1 Epoch

To find the source of significance within the representation effect, the least significant difference (LSD) procedure was used. The procedure conducts multiple pair-wise t-test comparisons of treatment means to categorize similarities and differences. Table 6-5 shows the groupings of means across input representations and subject reaction time profiles. The pair-wise relational model means were significantly different than means for relational and dimensional models. Thus, even after only one epoch of training, the model showed the ability to improve performance. Although the input representation effect was significant, the overall model was not, $Pr\{F(11,948)>1.49\}=0.1304$. Thus, to determine the full capability of the ANN models to learn subject order, more training was performed.

T tests (LSD) for variable: Offset_difference

NOTE: This test controls the type I comparisonwise error rate not the experimentwise error rate.

Alpha= 0.05 df= 948 MSE= 2.138199
Critical Value of T= 1.96
Least Significant Difference= 0.2269

Means with the same letter are not significantly different.

T Grouping	Mean	Observations	Representation
A	**0.4031**	**320**	**Pair-wise Rel.**
B	0.1094	320	Dimensional
B			
B	-0.0094	320	Relational

Alpha= 0.05 df= 948 MSE= 2.138199
Critical Value of T= 1.96
Least Significant Difference= 0.262

Means with the same letter are not significantly different.

T Grouping	Mean	Observations	Subject
A	0.2083	240	1
A			
A	0.1875	240	4
A			
A	0.1542	240	2
A			
A	0.1208	240	3

Table 6-5. LSD Analysis of Training on Subject Order for 1 Epoch

To insure the weights were trained (stabilized from major fluctuations), the models ANN_P to ANN_AA were run for 20000 epochs. The trained weights were then used by models ANN_D to ANN_O to predict performances for day 5 rectangle patterns. To test the effect of trained versus untrained weights, initial random weights were again used by models ANN_D to ANN_O to predict outcomes for day 5 patterns. Table 6-6 shows the offset differences for the models.

Subject	Representation: Dimensional	Representation: Relational	Representation: Pair-wise Relational
1	Model: ANN_D Mean of Difference*: 0.613	Model: ANN_E Mean of Difference: 0.612	Model: ANN_F Mean of Difference: 1.162
2	Model: ANN_G Mean of Difference: 0.325	Model: ANN_H Mean of Difference: 0.450	Model: ANN_I Mean of Difference: 1.012
3	Model: ANN_J Mean of Difference: 0.188	Model: ANN_K Mean of Difference: 0.750	Model: ANN_L Mean of Difference: 0.950
4	Model: ANN_M Mean of Difference: 0.112	Model: ANN_N Mean of Difference: 0.562	Model: ANN_O Mean of Difference: 1.112

* Mean of difference between results of model using untrained weights and model using trained weights.

Table 6-6. Artificial Neural Network Offset Results after 20000 Training Epoch on Subject Order

An ANOVA table was constructed to determine the significance of training for 20000 epochs in models ANN_D to ANN_O. Table 6-7 shows a two-way ANOVA with input representation and subject reaction time as

the effects. The ANOVA model is the same as *(6.1)*. The analysis found not only the representation to be significant, Pr{F(2,948)>18.43}=0.0001, but also the overall model to be significant, Pr{F(11,948)>3.99}=0.0001. Thus, training for 20000 epochs significantly improved learning from training for 1 epoch.

ANOVA Procedure

Dependent Variable: Offset_difference

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	**11**	**109.33333333**	**9.93939394**	**3.99**	**0.0001**
Error	948	2359.85000000	2.48929325		
Corrected Total	959	2469.18333333			

Source	DF	SS	Mean Square	F Value	Pr > F
Representation	**2**	**91.75208333**	**45.87604167**	**18.43**	**0.0001**
Subject	3	6.60000000	2.20000000	0.88	0.4490
Representation*Subject	6	10.98125000	1.83020833	0.74	0.6213

Table 6-7. Two-way ANOVA of Training on Subject Order for 20000 Epochs

Further investigation using the LSD procedure shows the importance of the type of input representation in determining a model's ability to learn. Table 6-8 shows that the means of differences were significant for each of the input representations. The pair-wise relational models learned most effectively with a 1.059 mean reduction of offset difference. Relational models did not learn as well with a 0.5938 mean reduction of offset difference. The dimensional model learned the least with a 0.3094 mean reduction of offset difference. Relatively, the pair-wise relational models were far superior in learning than models with the other two

representations. Perhaps the success of the pair-wise relational models could be attributed to their ability to capture configural feature information used by the subjects.

T tests (LSD) for variable: Offset_difference

NOTE: This test controls the type I comparisonwise error rate not the experimentwise error rate.

Alpha= 0.05 df= 948 MSE= 2.489293
Critical Value of T= 1.96
Least Significant Difference= 0.2448

Means with the same letter are not significantly different.

T Grouping	Mean	Observations	Representation
A	**1.0594**	**320**	**Pair-wise Rel.**
B	0.5938	320	Relational
C	0.3094	320	Dimensional

Alpha= 0.05 df= 948 MSE= 2.489293
Critical Value of T= 1.96
Least Significant Difference= 0.2827

Means with the same letter are not significantly different.

T Grouping	Mean	Observations	Subject
A	0.7958	240	1
A			
A	0.6292	240	3
A			
A	0.5958	240	2
A			
A	0.5958	240	4

Table 6-8. LSD Analysis of Training on Subject Order for 20000 Epochs

Weights Analysis for Pair-wise Relational Models

The significance of the pair-wise relational models raised an important question. Were the weights personalized to each subject? Or did the weights reveal diagnostic capabilities common to all subjects? To

answer the question, weights trained by a pair-wise relational model using one subject reaction time profile (ANN_R, ANN_U, ANN_X, or ANN_AA) were used to predict performances for models with a different reaction time profile. The configuration of the test was as follows:

Train on Model:	Predict on Models:
ANN_R	ANN_I, ANN_L, ANN_O
ANN_U	ANN_F, ANN_L, ANN_O
ANN_X	ANN_F, ANN_I, ANN_O
ANN_AA	ANN_F, ANN_I, ANN_L

Each training model was run for 20000 epochs on 240 rectangle patterns (days 2-4). The trained weights were then used by the prediction models to generate offset profiles for day 5 patterns. Table 6-9 shows the offset means per pattern for the 80 day-5 patterns. The right-hand column represents the average offset of 3 models, whereas the left-hand column shows one model's means offset.

Subject Profile for Prediction	Train on Same Subject Profile as Prediction Model	Train on Different Subject Profile as Prediction Model
1	Offset Mean: 2.138	Offset Mean: 2.117
2	Offset Mean: 2.000	Offset Mean: 2.196
3	Offset Mean: 2.175	Offset Mean: 2.150
4	Offset Mean: 2.138	Offset Mean: 2.138

Table 6-9. Artificial Neural Network Offset Results for Interchanged Weights

Comparisons between means in table 6-9 clearly show that a significant difference does not exist between the columns. Thus, weights trained on one subject reaction time profile could be used to predict on another profile without significant performance variations. Perhaps the trained weights for the pair-wise relational models carried diagnostic information common to all subjects.

Skeletonization

To unlock the success of the pair-wise ANN model, a technique was needed to look inside its workings. In particular, the technique should determine which input nodes were relevant and, therefore, attended by all models. Mozer and Smolensky (1989) developed a method, called skeletonization, to compute a measure of relevance which identifies critical input or hidden layer nodes and trims least relevant nodes. The skeletonization process is an iterative one. Backpropagation training is first performed to meet a performance criteria. Once met, the ANN model then trims the least relevant node. The process is repeated until the training criteria can no longer be met. The skeletonization procedure was adapted to the pair-wise relational ANN model in order to find a common diagnostic set of input nodes. Figure 6-6 shows the adaptation.

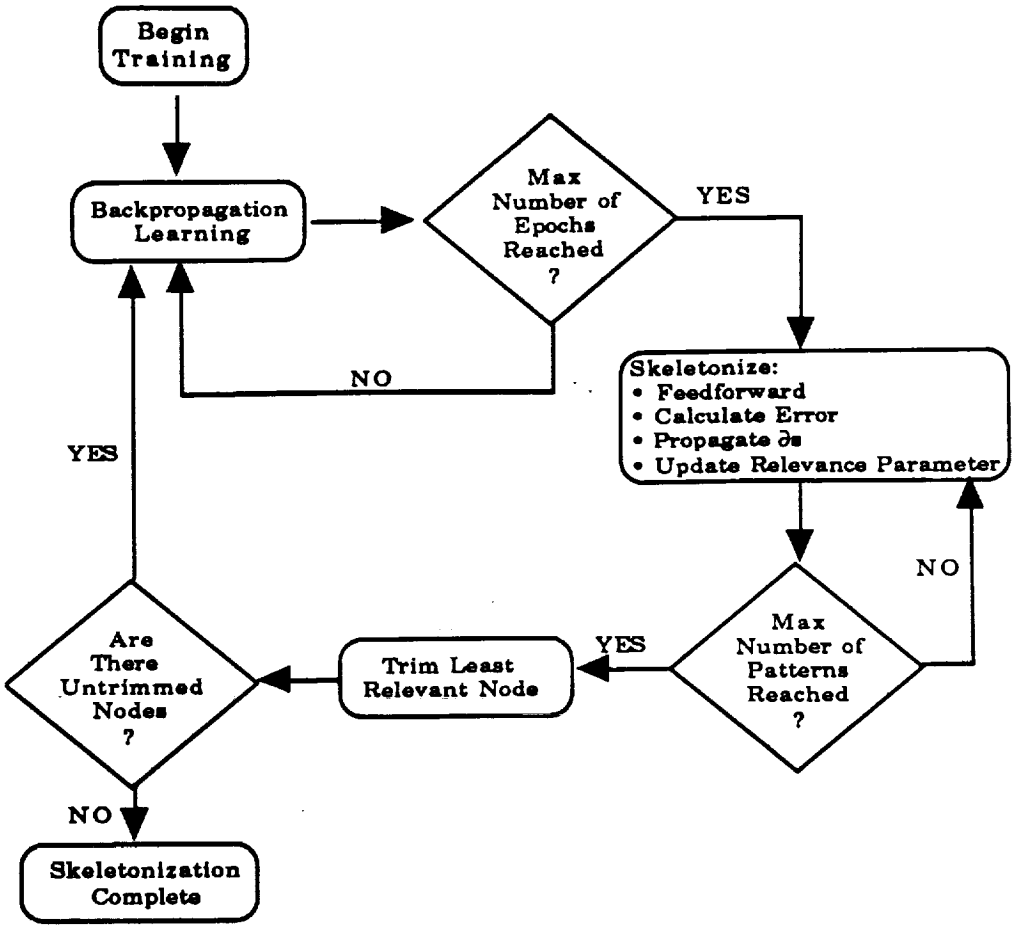


Figure 6-6. Skeletonization Flowchart

The adaptation consisted of two phases. The first phase involved training using the backpropagation algorithm. Once the maximum number of training epochs was reached (arbitrarily set at 100), the skeletonization procedure was executed. After the final input pattern was presented, and the relevance parameters updated, the least relevant input layer node was trimmed from the model. The relevance measure, P, of an input node, i, is calculated as follows:

$$P = \sum_{j} \partial(j) * W(j)(i) * X(i)$$

Where W represents weights from input node i to hidden node j and X represents input layer nodes. Once the least relevant node was trimmed, backpropagation learning occurred once again to train the reduced model. After training, the skeletonization procedure was again executed. This process continued until all input layer nodes had been trimmed. For a more detailed description of the skeletonization procedure, see Appendix 1.

Figures 6-7 and 6-8 show skeletonization results of the pair-wise relational models. Figure 6-7 depicts the trend of average payoff score at the final epoch before skeletonization. The overlay shows the trend of all four pair-wise relational models (ANN_F, ANN_I, ANN_L and ANN_O). Figure 6-8 shows the number of output layer nodes surpassing the error threshold (threshold < target node value - output layer node value) for the pair-wise relational models. As the number of trimmed nodes increased, the average payoff score decreased, and the number of nodes surpassing the threshold increased.

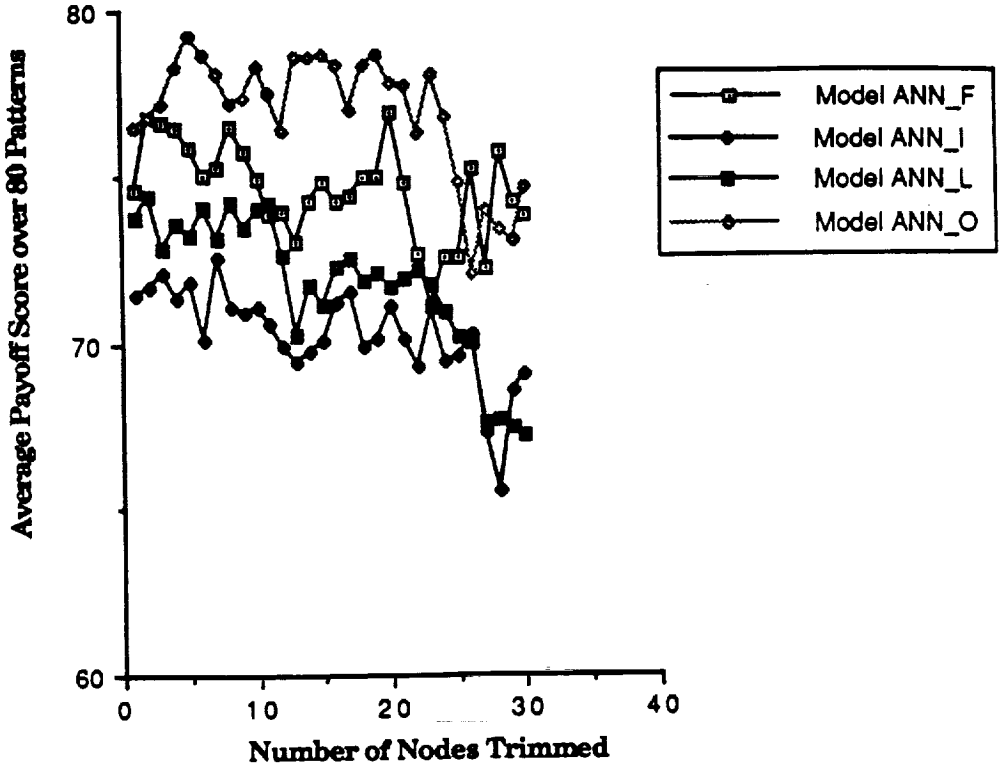


Figure 6-7. Overlay of ANN Payoff Performance Using Skeletonization

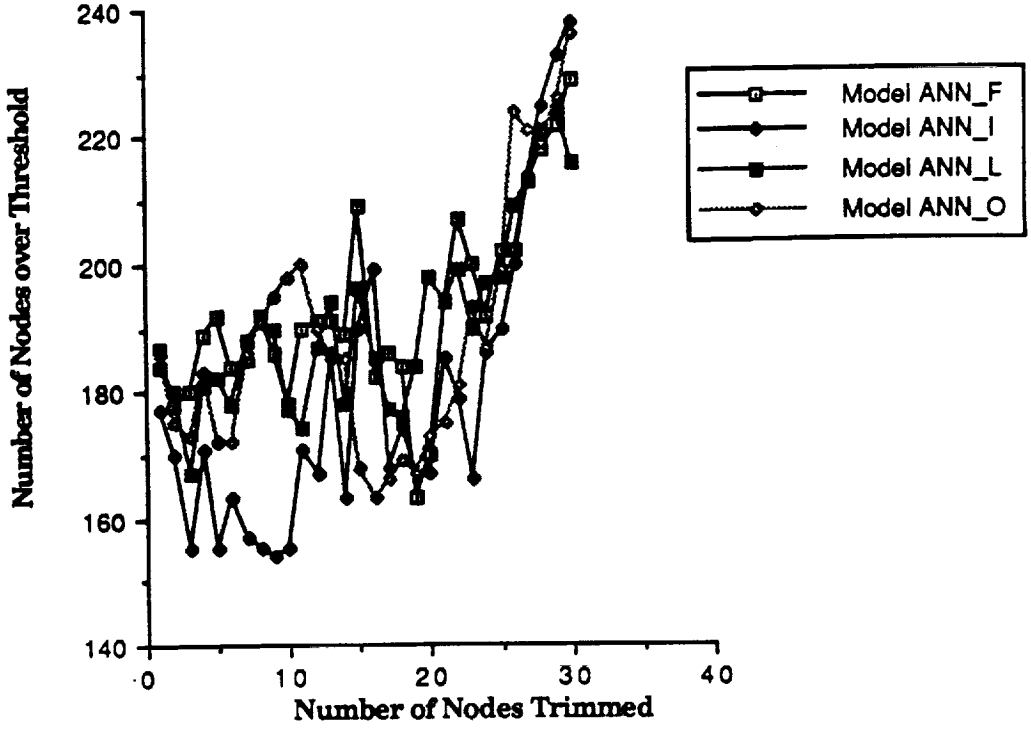


Figure 6-8. Overlay of ANN Offset Behavior Using Skeletonization

To reveal the importance of each input layer node, a profile of the trimmed nodes was made. By showing which nodes were relevant among all input layer nodes, a set of attended dimensional features in the RTT could be classified. Figure 6-9 shows a representation of the state of each of the 4 models with different subject reaction time profiles after 10 nodes have been trimmed. For each model, the remaining nodes are shown as circular numbered nodes. Each untrimmed pair-wise relational model has 30 input layer nodes divided into five feature representations as discussed in Chapter III. After 10 nodes have been trimmed, figure 6-9 shows 21 height (H) nodes, 19 width (W) nodes, 14 H*W nodes, 10 H+W nodes, and 16 H^2+W nodes remaining. Thus, after executing the skeletonization procedure, the height and width dimensions appeared to be more diagnostic than the others.

As a tool, the skeletonization procedure can analyze an ANN model for the most, as well as least, relevant nodes. Its application to the pair-wise relational model was not intended to give a detailed description of the worth of each input node. Rather, it was meant to give an overview of the content of the nodes with respect to the relevance of input features. In terms of the error signature discussed in Chapter II, the height order heuristic has a slope of 1 on the ln(height) X ln(width) graph. Since the optimal (H^2+W) heuristic has a slope of 1/2, the similarity between them makes height an attractive alternative that is perceptually available. Perhaps the skeletonization procedure was able to capture the diagnosticity of the height heuristic.

Model	Input Layer Node Identification	Input Representation Feature
ANN_F	1 2 3 4 6 7 8 9 10 11 12 14 15 16 18 22 25 26 28 30	H W H * W H + W H^2 + W
ANN_I	1 2 3 4 5 6 7 9 10 11 12 15 18 19 21 22 25 27 29 30	H W H * W H + W H^2 + W
ANN_L	2 3 4 5 8 9 10 11 12 13 15 16 18 19 21 22 25 26 29 30	H W H * W H + W H^2 + W
ANN_O	1 2 3 4 5 6 8 11 12 13 14 17 18 20 23 24 26 27 29 30	H W H * W H + W H^2 + W

Figure 6-9. Input Nodes Remaining after 10 Skeletonization Operations

GA Model Analysis

Before the GA models could be used to model subject performance, the models needed first to show the ability to learn the RTT. Thus, models GA_A, GA_B, GA_C, and GA_D were trained on the payoff scores generated by the optimal order for 1200 generations on 240 rectangle patterns (days 2-4). The payoff scores were average scores of all 50 rules in the population per each trial. The criteria of learning, as mentioned in Chapter 4, was the RTT payoff score. Unlike the ANN models, which had one measure of performance (one offset) per pattern, the genetic learning models had 50 measures (one per rule string in rule base) per generation. 1200 training generations were used to insure stabilization of the rule populations. The trained rule strings were then used by models GA_E, GA_F, GA_G, and GA_H to predict outcomes for displayed rectangle patterns on the final session (day 5). To test the effect of trained versus untrained rule strings, the initial, untrained rule strings were also used by models GA_E to GA_H to predict outcomes for day-5 patterns. Table 6-10 shows the difference between offsets generated by models using untrained and trained rule strings.

Subject	Model
1	Model: GA_E Mean of Difference: 15.138
2	Model: GA_F Mean of Difference: 77.638
3	Model: GA_G Mean of Difference: 34.512
4	Model: GA_H Mean of Difference: 58.888

Table 6-10. Genetic Algorithm Offset Results after 1200 Training Generations on Optimal Order

An ANOVA table was constructed to determine the significance of the offset differences in models GA_E to GA_H. Table 6-11 shows an one-way ANOVA with subject reaction time as the effect. The ANOVA model is as follows:

$$\text{Offset\_difference}_{mk} = \mu + \text{Subject}_m + \text{error}_{mk} \qquad (6.2)$$

for m=1,2,3,4; and k=1,...,80.

ANOVA Procedure

Dependent Variable: Offset_difference

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	180023.43750	60007.81250	21.50	0.0001
Error	316	881949.95000	2790.98085		
Corrected Total	319	1061973.38750			

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Subject	3	180023.43750	60007.81250	21.50	0.0001

Table 6-11. One-way ANOVA of Training on Optimal Order for 1200 Generations

The analysis found the subject reaction time profiles to be highly significant, Pr{F(3,316)>21.50}=0.0001. Thus, the amount of learning done by models GA_E to GA_H depended highly on the reaction times of the subjects. Nevertheless, all of the models showed performance improvements through training.

Analysis of GA Models of Subject Performance

Due to variations in subject reaction times, the GA learning criteria needed to change in order for GA models to simulate subject performance. To modify the criteria, a rewarding scheme based on the offset metric was established as follows:

- 0 Offset -> payoff score = 100
- 2 Offsets -> payoff score = 66
- 3 Offsets -> payoff score = 33
- 4 Offsets -> payoff score = 0

Thus, as the offsets are generated by subject and GA model ordering differences, an improvement in payoff performance will naturally reduce the number of offsets.

Using the revised learning criteria, the GA models were trained on subject performance. The GA rule population for models GA_I, GA_J, GA_K, and GA_L were trained on 240 patterns (days 2-4). The trained rule strings were then used by models GA_E, GA_F, GA_G, and GA_H to predict outcomes for displayed rectangle patterns on the final session (day 5). To test the effect of trained versus untrained rule strings, the initial, untrained populations were also used by models GA_E to GA_H to predict outcomes for day 5 patterns. The difference between offsets generated by models using untrained and trained rule strings were then contrasted across subject reaction times. Table 6-12 shows the mean of the offset differences per displayed rectangle pattern for each of the models. All of the models showed performance improvements through training.

Subject	Model
1	Model: GA_E Mean of Difference: 21.175
2	Model: GA_F Mean of Difference: 73.462
3	Model: GA_G Mean of Difference: 60.862
4	Model: GA_H Mean of Difference: 55.825

Table 6-12. Genetic Algorithm Offset Results on Subject Order

An ANOVA table was constructed to show the effects of training models GA_E to GA_H on subject order. Table 6-13 shows an one-way ANOVA with subject reaction time as the effect. The ANOVA model is the same as *(6.2)*. The analysis found, as in optimal order training results, the subject reaction time profiles to be highly significant, Pr{F(3,316)>13.24}=0.0001. Thus, although the models showed performance improvements, those improvements depended greatly on the reaction times of the subjects.

ANOVA Procedure

Dependent Variable: Offset_difference

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	120098.41250	40032.80417	13.24	0.0001
Error	316	955690.47500	3024.33695		
Corrected Total	319	1075788.88750			

Source	DF	SS	Mean Square	F Value	Pr > F
Subject	**3**	**120098.41250**	**40032.80417**	**13.24**	**0.0001**

Table 6-13. One-way ANOVA of Training on Subject Order

GBML Model Analysis

To determine whether the GBML models were capable of learning, the models were trained on 320 patterns (days 2-5) for 100 generations. The payoff profile generated by the models GBML_A, GBML_B, GBML_C, and GBML_D are shown in figures 6-10 to 6-13. The payoff score profile represents the average payoff score for the 50 rules in the population traced over 100 generations. After experimenting with different combinations of model parameters, the one with the following settings was chosen based on

performance: mutation probability set at 0.5; reward multiplier set at 1.1 times the bid; and the payoff threshold was set to geometrically increase from 55.0 to 90.0. Figure 6-10 to 6-13 shows that the GBML models are more volatile than previously expected. One rationale for the erratic behavior could be found in the mutation rate. Since earlier attempts of lower rates failed to improve performance, the rate was set at 0.5 to insure variability. However, with the increased rate, stability was sacrificed. Another reason for the model behavior could be found in the payoff threshold. As the threshold increased to a value not attainable by the models, performance became erratic and unpredictable.

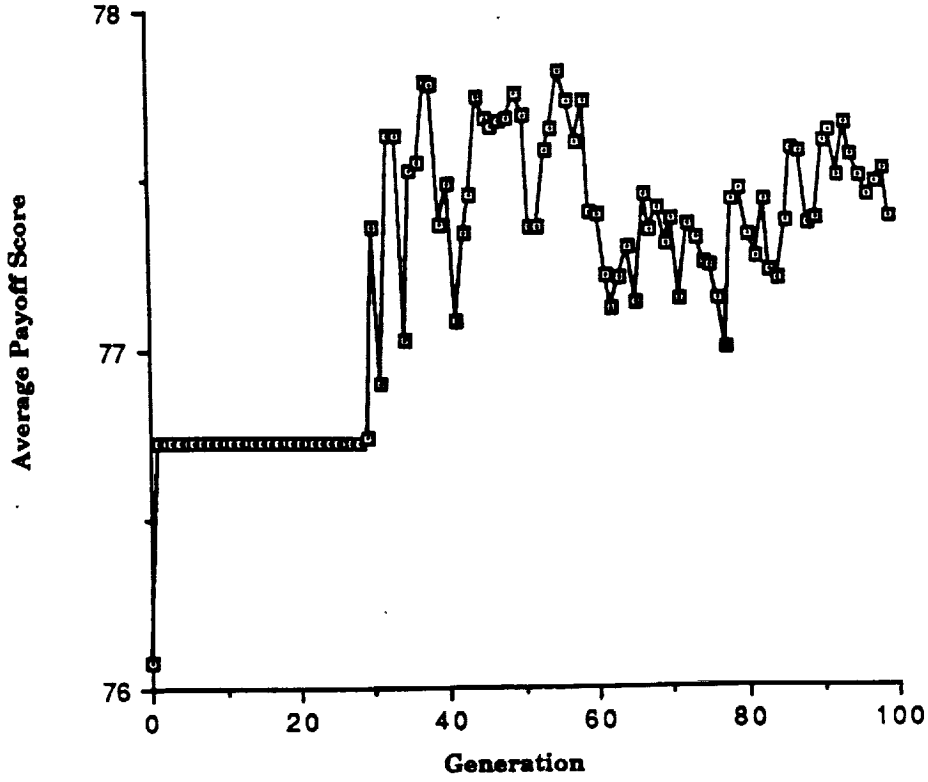


Figure 6-10. Model GBML_A Payoff Profile Using Subject 1 Reaction Times

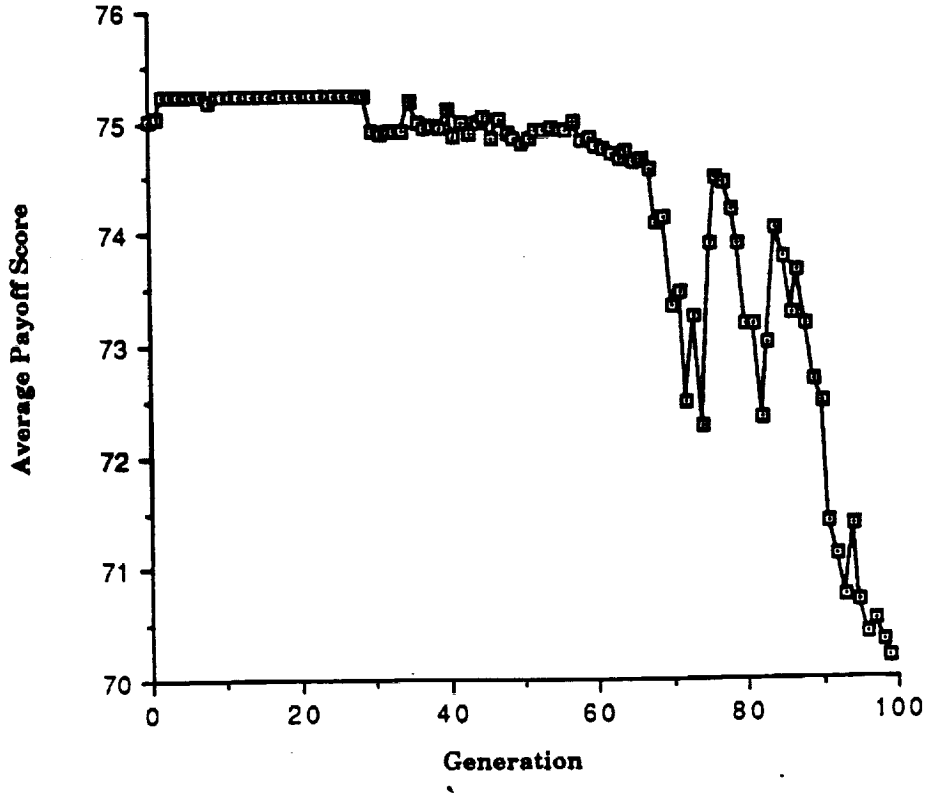


Figure 6-11. Model GBML_B Payoff Profile Using Subject 2 Reaction Times

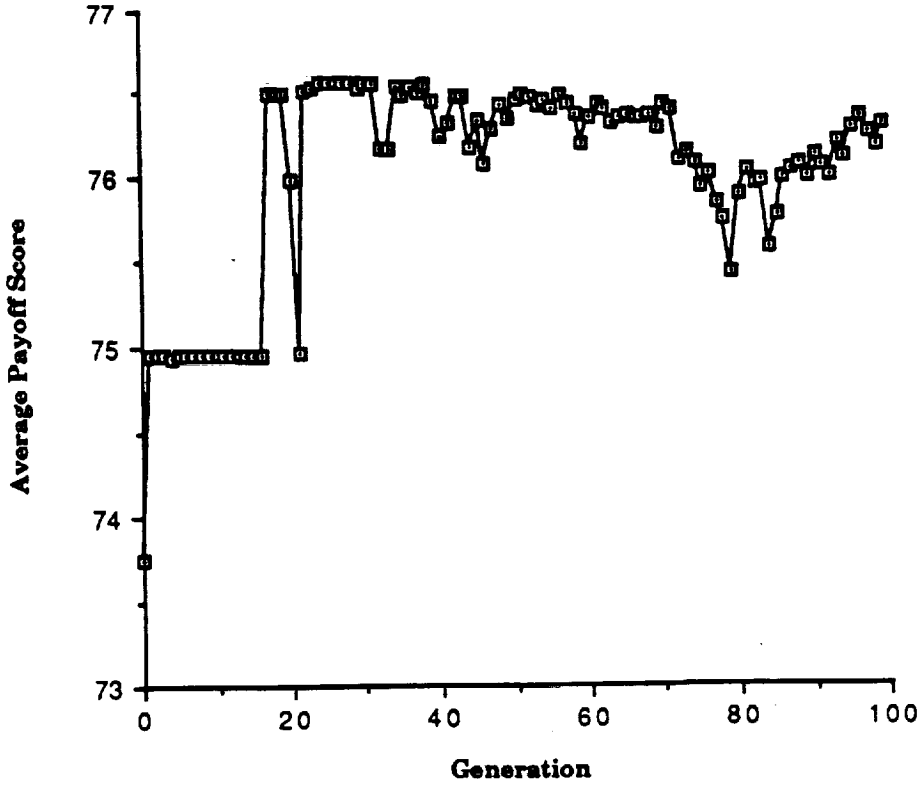


Figure 6-12. Model GBML_C Payoff Profile Using Subject 3 Reaction Times

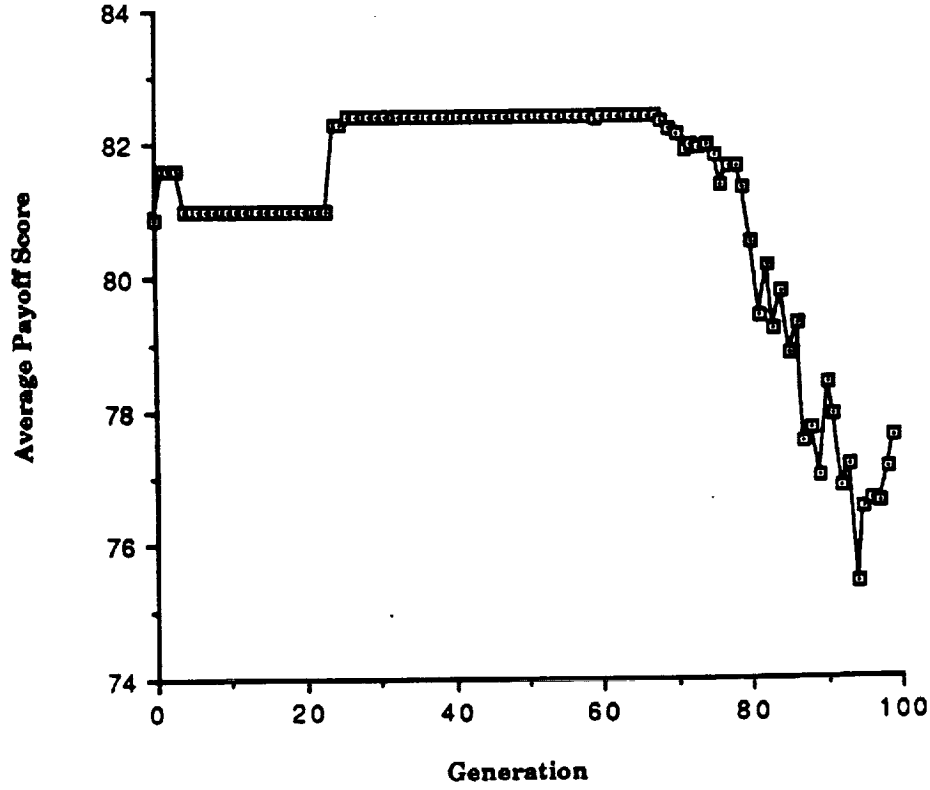


Figure 6-13. Model GBML_D Payoff Profile Using Subject 4 Reaction Times

Discussion of Results

Of all the models, the ANN structure shows the most promise. Through analysis of the learning abilities of the different representations, strategies of human decision-making could be inferred. The dimensional ANN models showed some learning abilities. Relational ANN model results show definite signs of improvement. Nevertheless, initial evaluations after 1 epoch indicate slow network responsiveness for both representations. The pair-wise relational ANN appears to be the most capable network to model skilled decision-making. Initial evaluations after 1 epoch indicate signs of learning. Training for 20000 epochs confirms initial observations. Skeletonization results suggest height and width cue utilization. However, the skeletonization process--the decision of when to trim the nodes and when the network has reached the minimal configuration--is crudely approximated from a graph, and should be considered only a rough approximation.

GA models also showed the ability to learn subject performance. However, since the resultant rule strings were the products of random search processes, the amount of learning varied widely from one model to the next. In addition, each rule string was unique, and did not reveal an accumulation of learning from past experience (as in the ANN weights). Thus, a skeletonization-like procedure did not exist for the GA models. Furthermore, the trained population of rule strings converged on one unique representation. The trained rule string for model GA_I interpreted to the heuristic H^2/W^3. The rule for model GA_J interpreted to the

heuristic H^3/W^3. The rule for models GA_K and GA_L was also translated to H^3/W^3. In spite of the lack of population diversity, the models produced heuristics similar to the optimal heuristic by training on subject orders.

GBML models were not able to learn subject performance with any consistency. In order for the model to learn, a priori knowledge of the payoff score range, with respect to different subject reaction times, was needed. Furthermore, the balance between stability and learning was difficult to reach. Therefore, too much subject-dependent information was required by the GBML model to merit the effort of continued modeling.

- Relational

The relational models showed better results than the dimensional counterparts. However, compared to the pair-wise relational model results, the relational ANNs performed poorly.

- Pair-wise Relational

The pair-wise relational results were the best of the three types of models. The success of the models suggests that subjects may attend to configural features to more than individual features. Thus, a person may see that rectangle A is taller than rectangle B rather than rectangle A is 2X mm tall and rectangle B is X mm tall. This supposition is consistent with the findings of Gluck, Bower, and Hee (1989).

In addition to diagnosticity, the internal structure of the ANNs could also be examined. An examination of the internal structure using the skeletonization procedure revealed the relevance of each node based on the structure of past input patterns. The relevant nodes denoted the attended input features. Assuming the ANN model attuned to the same set of input features as the subject. The relevant nodes of the ANN model, therefore, could also represent the features the subject considered most diagnostic to the RTT. A simplistic application of the skeletonization procedure showed height and width as the diagnostic features. However, the skeletonization procedure provided only a cursory look at feature diagnosticity.

GA models generated rule strings which successfully simulated subject performance. The fact that each of the rule string populations converged to one unique rule representation after training presented the following dilemma:

- Heuristic interpretation

In terms of the ln(height)Xln(width) graph, model GA_I generated a slope of 3/2, and models GA_J, GA_K, and GA_L all produced slopes of 1. Thus, models GA_J to GA_L all formulated heuristics similar to the optimal heuristic while training on subject order.

- System volatility

If subject performance could be described by a rule string, then the performance of the model could be excellent. However, if the rule string required a perceptually unavailable features--like 1/W^2--the model could be difficult to interpret psychologically.

The success of the GA models, therefore, can be attributed to a rule string representation which sufficiently describes subject performance.

Contrary to De Jong's (1990) characterizations of the classifier (GBML) approach, the behavior of the GBML models was chaotic. This behavior could be attributed to the following factors:

- Function of BBA not captured in MBBA

The bucket brigade algorithm contributes to the formation of default hierarchies of rules within a classifier system (Grefenstette, 1987). The default hierarchies are constructed through multiple transactions of each message among the rule strings. The link that is created between rule strings that have won bids to the same message defines a set of global rules. The stability of classifier systems is due largely to the existence of these links. Thus, as the GBML model of the RTT restricted trading of messages to only one transaction, the links (and the stability) did not exist.

- A priori information not available

Because a payoff threshold was required for the GBML models, knowledge of the range of the payoff score (in terms of the subject reaction times) was needed. Thus, unlike the ANN models, the GBML models required both reaction time and rectangle ordering information to learn the RTT.

Implications of Findings and Suggestions for Future Research

Words of caution must accompany the following observations. The RTT environment is relatively simplistic with few perceptual cues available to a subject. To model skilled human decision-making in a more complex task would require a thorough analysis of the environment. Findings of this research suggests the following:

- Concentrate on pair-wise relational representations

Results of the ANN models suggest that configural features are more informative than individual features. If the pair-wise relational models are indicative of human decision-making processes, then configural features in the environment should play a key role as input representations in future models. Rather than focussing on dimensional aspects of graphical features, perhaps the emphasis should be on relational characteristics (i.e. taller, smaller, larger, etc...).

- Seek rules to exploit consistencies in subject performance

Success of the GA models suggests that, given sufficient rule string and input representations, subject performance could be simulated by genetic algorithm techniques.

Future research in skilled decision-making should address upscaling current GA and ANN models to simulate human performance in more complex environments. In particular, the following issues need to be faced:

- Rule string selection

To use a GA model to simulate decision-making in complex settings, a perceptually available rule string representation is needed. Furthermore,

objective functions should generate values that relate with perceptually available quantities (i.e. shape or area).

- Input feature selection

Complex environments offer numerous cues as either primitive features, emergent features, or artifactual elements. To model human decision-making in such settings, a metric to select relevant features is needed.

Pao described the input selection issues as follows:

> The choice of features for the description of objects--which may be concepts or physical objects or situations or events--is a difficult but essential preprocessing task in the implementation of computer-based pattern recognition.
>
> To some extent, there is no right or wrong choice, as long as sufficient information has been included in the set of feature values. However, inappropriate choices lead to the need for complex decision rules or mappings ... whereas incisive choices result in simple and comprehensive rules.
>
> The task of determining the [feature representation] is a crucial part of implementing computer-based pattern recognition, and the problem remains with us in adaptive pattern recognition. Often, however, we have the opportunity to discover adaptively which of the [feature representation] features are important and which are irrelevant.
> (Pao, 1989, p. 9)

Input feature representations for ANN models described in this thesis did promote efficient model behavior. Perhaps the finding of the significance of pair-wise relational features could aid feature selection in the future.

APPENDIX 1

PSEUDO-CODE OF ARTIFICIAL NEURAL NETWORK MODEL OF PERCEPTUAL DECISION-MAKING

Note that all functions and files are enclosed by <>.

Program Description

The program is an artificial neural network with a 2-layer weights architecture and updating after each pattern.

Program Initialization

Program initialization takes place through reading files <input/bp_file> and <Tulga_file> and through user interface at execution. User interface prompts user to choose type of trace (per epoch or per pattern) and skeletonization. <input/bp_file> is read by <Backprop::initialize>. The following variables/constants are read by <Backprop::initialize>:

1) Number of rows of input signal pattern
2) Number of columns of input signal pattern
3) Number of rows of target pattern
4) Number of columns of target pattern
5) Number of signal patterns
6) Number of training epochs
7) Number of hidden layer elements
8) Starting learning rate. Learning rates are used for updating weights

   hidden_layer_weights = hidden_layer_weights + learning_rate * input_node_value * hidden_delta_value

   output_layer_weights = output_layer_weights + learning_rate * hidden_node_value * output_delta_value
9) Ending learning rate

10) Type of data (binary or bipolar)

11) Learning threshold (allowable difference between target and output)

12) Values of input and target slab nodes (target slab values initialized to 0)

13) Values of input and output layer weights (random value x such that -1< x < 1) read from <input/weight_file>

The following variables/constants are calculated by <Backprop::initialize>:

1) Learning rate multiplier (geometric progression based on total number of training patterns)

2) Input vector length (number of rows * number of columns)

3) Target vector length (number of rows * number of columns)

4) Relevance toggles (initialized to 0)

<Tulga_file> is read by <Tulga::init_Tulga>. In addition, files <output/ordering>, <output/payoffs>, and <output/pattern_tr> are initialized and replace old files. The following variables/constants are read by <Tulga::init_Tulga>:

1) Identification number of boxes to be processed

2) Subject reaction time

The following variables/constants are calculated by <Tulga::init_Tulga>:

1) Epoch counter (initialized to 1)

2) Pattern counter (initialized to 1)

3) Payoff score, average, sum, threshold limit, and standard deviation (initialized to 0)

Program Execution

The program contains a main loop which executes one of the following three subroutines:

1) Backpropagation learning

2) Feedforward

3) Skeletonization

Nested within the main loop is a secondary loop composing the first subroutine. The secondary loop consists of the following elements:

1) <Tulga::feedforward> executes <Backprop::FFhidden> and <Backprop::FFoutput>

<Backprop::FFhidden> sums hidden slab node inputs as follows:

hidden_nodes_sum_inputs = hidden_nodes_sum_inputs + hidden_weights_strength * input_nodes_value

<Backprop::FFhidden> also calculates the hidden slab node activations using the binary logistic sigmoid function

<Backprop::FFoutput> sums output slab node inputs as follows:

output_nodes_sum_inputs = output_nodes_sum_inputs + output_weights_strength * hidden_nodes_activation

<Backprop::FFoutput> also calculates the output slab node activations using the binary logistic sigmoid function

2) <Tulga::cognition> calculates cardinality of output_nodes_activations (strength order), converts strength order to selection order, writes selection order to file <output/ordering>, calculates payoff score and

optimal order, converts optimal order to optimal strength order, sets target vector elements based on strength order, and calculates over_thresh (number of output_nodes_activations surpassing allowed threshold from target_nodes_value)

3) <Tulga::backpropagation> executes <Backprop::BPoutput> and <Backprop::BPhidden>

<Backprop::BPoutput> calculates error values as follows:

error_value = target_nodes_value - output_nodes_activations

<Backprop::BPoutput> calculates the output layer delta values as follows:

output_delta_value=

logistic_sigmoid_derivative(output_nodes_activations) *

error_value

<Backprop::BPhidden> calculates the hidden layer delta values as follows:

hidden_delta_value =

logistic_sigmoid_derivative(hidden_nodes_activations) *

Σ (output_layer_weights * output_delta_value)

4) <Tulga::update_weights> updates weights using the delta rule and executes subroutines based on following conditionals:

if epoch counter is equal to the maximum number of training epochs,

write message to file <output/payoffs>, save hidden and output layer weight strengths to file <output/new_weights>, if per epoch trace is on, calculate payoff statistics and save results to

file <output/payoffs>, and return 0 as current status

if pattern counter is equal to maximum number of patterns and the number of error terms less than the threshold is within limits, write message to file <output/payoffs>, save hidden and output layer weight strengths to file <output/new_weights>, if per epoch trace is on, calculate payoff statistics and save results to file <output/payoffs>, and return 0 as current status

if pattern counter is equal to maximum number of patterns, if per epoch trace is on, calculate payoff statistics and save results to file <output/payoffs> else save current payoff and threshold information to file <output/pattern_tr>, reset pattern counter to 1, increment epoch counter, set threshold counter to 0, return 1 as current status

if none of the above conditions hold, if per epoch trace is off, save current payoff and threshold information to file <output/pattern_tr>, increment pattern counter, and return 1 as current status

if current status is 0, the program exits the secondary loop

If the feedforward option is selected, the following loop is executed for the maximum number of epochs:

1) <Tulga::feedforward> (as explained in secondary loop description)

2) <Tulga::cognition> (as explained in secondary loop description)

If the skeletonization option is selected, the following loop is executed:

1) Backpropagation Learning (as explained in secondary loop description) executes for the maximum number of training epochs.

2) <Backprop::FFhidden> (as explained in secondary loop description)

3) <Backprop::FFoutput> (as explained in secondary loop description)

4) <Tulga::cognition> (as explained in secondary loop description)

5) <Backprop::SKoutput> sets the error values as follows:

    if output_nodes_activations is greater than target_nodes_value, set error_value to -1.0 else set error_value to 1.0 next calculate the output layer delta values based on the errors

6) <Backprop::SKhidden> calculates the hidden layer delta values and the input node relevance values as follows:

    input_nodes_relevance = 0.2 * input_nodes_value * Σ (hidden_layer_weights*hidden_delta_value) + 0.8 * input_nodes_relevance

    Note: constants are based on paper by Mozer and Smolensky (1989).

To complete the skeletonization option, <Backprop::SKremove> and <Tulga::reset> are executed and a toggle is returned to determine continuation of program. <Backprop::SKremove> calculates the irrelevant node (node with smallest relevance value), writes relevance value of all nodes to file <output/payoffs>, sets input_nodes_value of irrelevant node to 0.0, sets all weights emanating from the irrelevant node to 0.0, reset all input_nodes_relevance to 0.0, and sets toggle to 0 if all nodes have been

"trimmed". If toggle is set to 0, skeletonization terminates. <Tulga::reset> resets epoch and pattern counters to 1, payoff average, payoff sum, threshold limit and standard deviation of payoffs to 0, and resets learning rate to value specified initially in file <input/bp_file>.

APPENDIX 2

PSEUDO-CODE OF GENETIC MODEL OF PERCEPTUAL DECISION-MAKING

Program Description

The program contains options to execute either a pure genetic algorithm or a GBML.

Program Initialization

Program initialization takes place through reading files <input/rulefile> and <input/rec_file> and through user interface at execution. User interface prompts user to choose either a pure GA or GBML and the number of generation to execute. Files <output/trace> (trace of rules at each generation) and <output/rule_strength> (strength of each rule at each generation) are initialized. The following variables/constants are read by <Cs::rule_init> from file <input/rulefile>:

1) Random number seed
2) Upper limit of random number sequence
3) Lower limit of random number sequence
4) Probability of mutation in the genetic algorithm (string will mutate if probability is lower than stated)
5) Bid percentage (for GBML) in decimal form
6) Reward (for GBML) as percentage (in decimal form) of bid
7) Begin payoff threshold (for variable threshold in a GBML)
8) End payoff threshold (for variable threshold in a GBML)
9) Length of rule string
10) Number of strings in population
11) Number of components in rule string (ex. in a+b there are three components)

12) Rule string bits

13) Original rule strength (for GBML)

The following variables/constants are read by <Cs::rule_init> from file <input/rec_file>:

1) Number of rectangle sets

2) Identification number of boxes to be processed

3) Height and width of boxes

4) Subject reaction time

In addition to reading variables and constants, the following subroutines are executed by <Cs::rule_init>:

1) <Random::initialize_random> sets the seed, upper and lower limits of pseudo-random number generator

2) <Genetic::set_mutate_probability> sets the genetic algorithm mutation probability

3) <Cs::set_payoff_information> sets the bid percentage, reward percentage, and payoff threshold. Also calculates the threshold multiplier (uses geometric progression based on total number of generations)

Program Execution

If the pure GA is chosen, the following functions are executed for number_of_generations * number_of_patterns times:

1) <Cs::pureGA> executes <Cs::translate>, rule_strength, and offsets from optimal order for all rule strings

<Cs::translate> uses the current rule interpretation:

h ^a (operators) w ^b

where h is height of rectangle, w is width of rectangle, and operators are +, -, *, /.

— — —

aa bb operators

ex. 10 01 111 = h^2 / w

based on the rule used, a value is recorded for each rectangle and ordered rule_strength represents the payoff score calculated given the ordering provided by <Cs::translate> offsets are calculated by comparing the optimal order and the <Cs::translate> order

2) <Cs::print_rules> writes rules of the current generation to file <output/trace> and rule strength of all rule strings (as well as the average rule strength) to file <output/rule_strength>

3) <Genetic::reproduce> reproduce performs the reproductive function within the genetic algorithm. It takes the strongest (probabilistically) rule to replace the weakest rule. Each rule is given a slot on the "roulette wheel" based on the strength. The wheel is spun n times for n rule strings in the population. The new population then replaces the old.

4) <Genetic::crossover> performs "mating" between rules. It identifies the number of pairs in population, pairs available rule mates, selects crossover site, and crosses bit information between pairs (this portion of code signifies a major departure from conventional crossover

procedures in that only a rule segment is crossed between mates)

5) <Genetic::mutate> mutate changes at most one rule string at random given a probability of change. If the pseudo-random number generated is less than the mutate probability, a mutation sector (i.e. rule components) is randomly generated and changed (if random number is greater than .5 set binary allel to 1 otherwise, set to 0)

If the GBML is chosen, the following functions are executed for number_of_generations times:

1) <Cs::aoc> is the apportionment of credit system. Must be given input rectangles and time. For the number of messages (usually number of rule strings in population), <Cs::aoc> calculates the maximum bid (rule_strength * bid_percentage), pays the bid from winning rule string, <Cs::translate> translates winning rule and return order, calculates payoff score (for statistical reasons), calculates the number of offsets of <Cs::translate> order from optimal order, and reward (positive or negative based on payoff threshold) the winning rule string (rule_strength * reward_percentage)

2) <Cs::print_rules> (as explained in pure GA description)

3) <Cs::stats> calculates payoff average and standard deviation for all rectangle patterns in one CS generation.

4) <Cs::savePay> records current CS generation, payoff average, standard deviation, and offset to file <output/payoffs>

5) <Genetic::reproduce> (as explained in pure GA description)

6) <Genetic::crossover> (as explained in pure GA description)

7) <Genetic::mutate> (as explained in pure GA description)

8) <Cs::reset_strength> resets all rule string strengths to the original strength and modifies the payoff threshold (payoff_threshold = multiplier * payoff_threshold)

REFERENCES

Axelrod, R. (1985). The Simulation of Genetics and Evolution. Paper Presented at a Conference on Evolutionary Theory in Biology and Economics, University of Bielefeld, Federal Republic of Germany.

Barnett, B.J. and Wickens, C.D. (1988). Display Proximity in Multicue Information Integration: The Benefit of Boxes. *Human Factors*, 30(1), 15-24.

Caudill, M. (1987). Neural Networks Primer: Part I. *AI Expert*, pp. 46-61.

Caudill, M. (1988). Neural Networks Primer: Part III. *AI Expert*, pp. 53-67.

Chase, W.G. and Simon, H.A. Perception in chess. *Cognitive Psychology*, 4, 1973.

Davis, L.H. (1985). Prisoners, Paradox, and Rationality. In *Paradoxes of Rationality and Cooperation*, R. Campbell and L. Sowden, Eds. Vancouver: University of British Columbia Press.

De Jong, K.A. (1990). Genetic-Algorithm-Based Learning. *Machine Learning Volume III*, Y. Kodratoff and R. Michalski (Eds.), ch. 21, 611-638.

Fausett, L.V. (1992). *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley Publishing.

Gluck, M.A., Bower, G.H., and M.R. Hee. (1989). A Configural-cue Network Model of Animal and Human Associative Learning. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI.

Grefenstette, J.J. (1987). Multilevel Credit Assignment in a Genetic Learning System. *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA.

Grefenstette, J.J. (1990). Genetic Algorithms and Their Applications. *The Encyclopedia of Computer Science and Technology*, Vol 21, A. Kent and J.G. Williams (Eds.), Marcel Dekker.

Hammond, K.R., Hamm, R.M., Grassia, J., and Pearson, T. (1987). Direct Comparison of the Efficacy of Intuitive and Analytical Cognition in Expert Judgment. *IEEE Transaction on Systems, Man, and Cybernetics*, Vol. SMC-17, No. 5.

Kirlik, A. (1992). *Requirements for Psychological Models to Support Design: Towards Ecological Task Analysis.* (In Review).

Kirlik, A., Markert, W.J., and R.J. Shively. (1990). Perceptual and Contextual Influences on Dynamic Decision-making Performance. *Proceedings of the 1990 IEEE International Conference on Systems, Man,and Cybernetics*, Los Angeles, CA.

Kirlik, A., Miller, R.A., and R.J. Jagacinski. (in press). Supervisory Control in a Dynamic and Uncertain Environment II: A Process Model of Skilled Human-Environment Interaction. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 22 no. 4.

Klein, G. (1989). Recognition-primed decisions. In *Advances in Man-Machine Systems Research*, Vol. 5, W.B. Rouse, Ed. JAI Press, Greenwich, CT.

Lesgold, A., Rubinson, H., Feltovich, P., Glaser, R., Klopfer, D., and Wang, Y. (1988). Expertise in a Complex Skill: Diagnosing X-ray Pictures. In *The Nature of Expertise.* M. Chi, R. Glaser and M. Farr (Eds). Lawrence Erlbaum Associates, Hillsdale, NJ.

Leven, S. and Elsberry, W. (1990). Interactions among Embedded Extensive Networks under Uncertainty. *Proceedings of the 1990 International Joint Conference on Neural Networks*. San Diego, CA.

Lippmann, R.P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, pp. 4-22.

Mozer, M.C. and P. Smolensky. (1989). Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. In *Advances in Neural Information Processing Systems 1*, D.S. Touretzky, Ed. Morgan Kaufmann Publishers, Palo Alto, CA.

Pao, Y.H. (1989). *Adaptive Pattern Recognition and Neural Networks.* Reading: Addison-Wesley Publishing.

Reynolds, R. G. (1979). An Adaptive Computer Model of the Evolution of Agriculture for Hunter-gatherers in the Valley of Oaxaca, Mexico. Unpublished Doctoral Dissertation, University of Michigan, Ann Arbor.

Richards, W., Ed. (1990). Representations in Mental Models. Final Technical Report. AFOSR.

Riolo, R.L. (1989). The Emergence of Default Hierarchies in Learning Classifier Systems. *Proceedings of the Third International Conference on Genetic Algorithms*. George Mason University, Virginia.

Rumelhart, D.E., Smolensky, P., McClelland, J.L., and G.E. Hinton. (1986). Schemata and Sequential Thought Processes in PDP Models. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2: Foundations. MIT Press.

Rumelhart, D.E., Hinton, G.E., and J.L. McClelland. (1986). A General Framework for Parallel Distributed Processing. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: Foundations. MIT Press.

Rumelhart, D.E., Hinton, G.E., and R.J. Williams. (1986). Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: Foundations. MIT Press.

Sanderson, P.M., Flach, J.M., Buttigieg, M.A., and Casey, E.J. (1989). Object Displays Do Not Always Support Better Integrated Task Performance. *Human Factors*, 31(2), 183-198.

Schneider, W. and Detweiler, M. (1988). The Role of Practice in Dual-Task Performance: Toward Workload Modeling in a Connectionist/Control Architecture. *Human Factors*, 30(5), 539-566.

Schuurmans, D., Chai, R., and L. Shu. (1987). Learning Using Classifier Systems: A Survey. *Proceedings of the 1987 Canadian Information Processing Society Conference*, Edmonton, Canada.

Simon, H.A. (1959). Theories of Decision-making in Economics and Behavior Science. *American Economic Review*, 49: 253-280.

Spiessens, P. (1990). PCS: A Classifier System that Builds a Predictive Internal World Model. *Proceedings of the 9th European Conference on Artificial Intelligence*. Stockholm, Sweden.

Tulga, M.K. and Sheridan, T.B. (1980). Dynamic Decisions and Workload in Multitask Supervisory Control. *IEEE Transaction on Systems, Man, and Cybernetics*, SMC-10.

von Neumann, J. and O. Morgenstern. (1947). *Theory of Games and Economic Behavior* (2nd Ed.). Princeton University Press, Princeton, NJ.